Tiago António Elias da Silva Freitas

[Licenciatura em Engenharia Eletrotécnica e de Computadores]

Estabilização de vídeos com base em descritores H.264

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: Prof. Doutor José Manuel Matos Ribeiro da Fonseca, FCT-UNL

Coorientador: Prof. Doutor Alexandre José Malheiro Bernardino, IST-UTL

Júri:

Presidente: Prof. Doutor André Teixeira Bento Damas Mora, FCT-UNL

Arguente(s): Prof. Doutor Arnaldo Joaquim Castro Abrantes, ISEL

Vogal(ais): Prof. Doutor Alexandre José Malheiro Bernardino, IST-UNL

Prof. Doutor José Manuel Matos Ribeiro da Fonseca, FCT-UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março de 2013

COPYRIGHT

AGRADECIMENTOS

Ao Professor José Manuel Fonseca, meu orientador da tese, pelo apoio e interesse, pela oportunidade cedida, pela confiança depositada na realização deste projeto e pelas palavras sempre bem-dispostas em qualquer momento.

Ao Professor Alexandre Bernardino, do Instituto Superior Técnico (IST) e meu coorientador da tese, o meu maior agradecimento pelo apoio incansável, pelos ensinamentos e partilha de conhecimentos, pela paciência e pela sua inteira disponibilidade em ajudar-me a realizar este projeto com o maior sucesso.

Ao Bernardo Motta e ao Pedro Soares o maior agradecimento pelo apoio e pela oportunidade única de realizar um projeto tão aliciante como este e produzir a minha tese de mestrado numa empresa tão distinta como a Observit.

Aos meus pais e família pelo incentivo, pelo apoio, pela oportunidade de poder frequentar este ensino e por confiarem e acreditarem sempre no meu trabalho. Espero ser sempre merecedor dessa admiração. Em especial ao meu pai pela disponibilidade, ensinamento e partilha de conhecimentos sempre tão úteis e grandiosos ao longo destes anos, mas sobretudo nos últimos meses.

À Mónica Lamas um agradecimento único pelo carinho, pelo sorriso, pela paciência, pelo incentivo incansável e apoio incondicional, por acreditar sempre, pelo orgulho e amizade, pelas palavras sempre especiais e pela presença insubstituível em todos os momentos.

Ao Carlos Valente o meu especial agradecimento pela companhia e conversas, pelo compromisso, pelas palavras sempre incentivantes, pela Amizade e pelo apoio diário todas as manhãs e fins de tarde.

A toda a equipa da Observit pela amizade, pelos momentos de descontração, pela ajuda essencial sempre que precisei e por me receberem tão bem nesta empresa.

Aos meus Amigos e colegas que tanto me ajudaram e acompanharam durante estes 5 anos de faculdade e a todos os que contribuíram para que este rumo tenha sido possível, o meu maior Obrigado.

RESUMO

No decorrer dos últimos anos, a qualidade de imagens e vídeos em câmaras tem evoluído, tendo a estabilização digital de imagem vindo a desempenhar um papel importante para a obtenção de vídeos estáveis em variadas situações, nomeadamente, vídeos amadores (*shaky movies*) ou determinadas áreas de videovigilância. Neste tipo de processamento existem vários métodos de estabilização, desde a vertente 2D até à mais complexa em 3D. O trabalho que aqui se apresenta pretende aplicar modelos de estabilização em áreas de videovigilância, no entanto, mostra-se igualmente bastante eficaz para qualquer tipo de vídeo amador.

O trabalho aqui desenvolvido demonstra um método construído em 2D baseado na estimação robusta de homografias abrangendo quatro modelos distintos de transformação de imagem: os modelos translacional, euclidiano, afim e projetivo, apresentados por ordem de complexidade. Estes quatro modelos distinguem-se sobretudo nos níveis de estabilização que se pretende aplicar a um vídeo. Ou seja, parte-se do princípio que quanto maior o número de parâmetros a estabilizar, mais complexo deverá ser o modelo aplicado. O modelo translacional pretende estabilizar apenas os movimentos indesejados nos eixos horizontal e vertical; o modelo euclidiano pretende estabilizar, para além destes, os movimentos rotacionais indesejados; o modelo afim introduz uma complexidade muito maior em termos de parâmetros relativamente aos anteriores, estabilizando, para além dos mencionados anteriormente, também os efeitos de escalamento, compressão e distorção de objetos; por último, o modelo projetivo pretende acrescentar aos anteriores a eliminação de perspetiva horizontal e/ou vertical existente nas imagens.

O método desenvolvido extrai os *keypoints frame* a *frame* comparando a posição de cada um em *frames* consecutivas, calculando assim a homografia inversa aplicável às imagens em cada modelo. Tendo isto em conta, a estabilização digital de imagens pode tornar-se, na visão por computador, num dos processamentos mais lentos e exigentes a nível computacional quando se enfrentam qualidades de vídeo bastante elevadas. Desta forma, para além do método de estabilização desenvolvido, este trabalho vem apresentar uma forma eficaz de aceder aos descritores visuais dos vídeos comprimidos em H.264 e extrair a informação neles presentes, acelerando assim todo o processo de estabilização.

Palavras-chave: estabilização de vídeo, estimação robusta, homografia, *keypoints*, *frames*, H.264

ABSTRACT

Throughout the last years, the quality of images in digital cameras has improved significantly, and the digital stabilization of images is playing an important role to achieve stable video under several conditions, such as amateur video (*shaky movies*), or certain surveillance systems. Several methods of image stabilization are used to process image data, from simpler 2D through the most complex algorithm in 3D. This study aims to develop stabilization models to be applied to surveillance video areas, however it shows to be quite effective when applied to any amateur video file as well.

This work produced one method developed in 2D concept based on a robust homographic estimation considering four different models of image data transformation: the models translational, rotational, affine and projective are presented in an increasing level of complexity, and they depend on the level of stabilization required for the final video, assuming that the higher the number of parameters to stabilize, the more complex the model will be. The translational model aims to stabilize only the undesired movements in x and y axes; the Euclidean model adds stabilization for the undesired rotational movements; the affine model is significantly more complex by adding stabilization to undesired scaling, compression or distortion of objects; and the projective model additionally eliminates the horizontal and vertical perspective in each image.

The developed methodology extracts the keypoints in sequential frames and compares each keypoint in consecutives frames, leading to the calculation of the reverse homographic to be applied to each image. Considering the complexity and the amount of data to process, the digital image stabilization turns into a very slow and highly demanding computer process, especially when dealing with high definition videos. Besides the development of the stabilization algorithms, this work presents an effective way to manipulate the motion vectors of H.264 compressed videos in order to extract the necessary information to accelerate the whole stabilization process.

Key words: video stabilization, robust estimation, homographic, keypoints, frames, H.264

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIATURAS

ACRÓNIMOS

FCT	Faculdade de Ciências e Tecnologias
UNL	Universidade Nova de Lisboa
IST	Instituto Superior Técnico
UTL	Universidade Técnica de Lisboa
HDA	*High Definition Analytics*
QREN	Quadro de Referência Estratégica Nacional
CCD	*Charge-Coupled Device*
OSIS	*Optical Shift Image Stabilizer*
VAP-IS	*Vari-Angle Prism Image Stabilizer*
GM	*Global Motion*
ME	*Motion Estimation*
MB	Macro-Blocos
MV	*Motion Vectors* (Vetores de Movimento)
LMV	*Local Motion Vectors*
RANSAC	*RANdom SAmple Consensus*
DLT	*Direct Linear Transformation*
SVD	*Singular Value Decomposition*
AVC	*Advanced Video Coding*
MPEG	*Moving Picture Experts Group*
FFMPEG	*Fast Forward* MPEG
CPU	*Central Processing Unit*
GPU	*Graphic Processing Unit*
CUDA	*Compute Unified Device Architecture*
Qpel	*Quarter-pixel*
STE	*Symmetric Transfer Error*

1. OBJETIVO

O principal objetivo deste projeto incide na construção de um sistema de estabilização de vídeo capaz de compensar em tempo real os deslocamentos de cada *frame* em relação à sua posição inicial (ao longo do tempo). Num sistema de estabilização de vídeo as perturbações são causadas essencialmente por variações externas à câmara [1]. Ou seja, a estabilização vem reduzir ou remover qualquer movimento indesejado presente numa sequência de imagens capturadas para que posteriores processamentos sobre o vídeo sejam os mais corretos e apropriados [2]. Esta é uma subárea do processamento digital de vídeo e surge com a necessidade de melhorar, de forma significativa, (isto é, proporcionar vídeos estáveis e sem perturbações) a qualidade de vídeo obtida bem como a eficiência de todo o processamento de *Video Analytics*.

É, então, com este intuito que se analisa e desenvolve a estabilização de vídeo descrita nesta dissertação. Nela dá-se a conhecer o desenvolvimento de técnicas e algoritmos de estabilização de imagens recorrendo a métodos de estimação robusta de homografias, com o objetivo de corrigir instabilidades ou movimentos indesejáveis presentes em vídeos extraídos de câmaras de segurança. Por outro lado, e com o objetivo de acelerar este processo, recorre-se à utilização de uma Unidade de Processamento Gráfico (ou *Grafic Processing Unit* - GPU, do inglês) para reduzir os tempos de processamento associados a funções pesadas computacionalmente. Numa segunda fase, aproveitando todas as vantagens inerentes às novas tecnologias de compressão de vídeo em HD, nomeadamente existentes no tipo de compressão H.264/AVC, extraem-se os descritores de movimento que podem ser diretamente aplicados no algoritmo de estabilização desenvolvido na primeira fase deste projeto com o mesmo objetivo da aplicação de uma GPU, mas de forma ainda mais eficiente do ponto de vista de rapidez de processamento. Por ser um processo bastante demorado e com a relevância demonstrada, apresenta-se aqui uma forma eficiente e eficaz de reduzir estes tempos contribuindo-se para a evolução das novas tecnologias utilizadas no mercado.

2. INTRODUÇÃO

O trabalho que se apresenta nesta dissertação foi realizado no âmbito da Unidade Curricular Dissertação do 2º ano de Mestrado em Engenharia Eletrotécnica e de Computadores (EEC) da Faculdade de Ciências e Tecnologias (FCT), da Universidade Nova de Lisboa (UNL) para obtenção do grau de Mestre em EEC. Foi inteiramente desenvolvido na empresa Observit, inserido num projeto denominado de *High Definition Analytics* (HDA), em colaboração com o Instituto Superior Técnico (IST) da Universidade Técnica de Lisboa (UTL).

O HDA é um projeto Quadro de Referência Estratégica Nacional (QREN) de pesquisa e desenvolvimento que surgiu com o objetivo de desenvolver novos algoritmos e métodos para suportar importantes desafios existentes na análise de vídeo, num contexto de novas tecnologias presentes em câmaras IP disponíveis no mercado: câmaras de alta resolução multi-megapixel e omnidirecionais[1].

Nos últimos anos tem sido visível um crescimento significativo de vídeos capturados por câmaras amadoras. Basta aceder à Internet para se deparar com filmagens em qualquer tipo de situações (e.g. na prática de desportos radicais) e a partilha constante destes mesmos conteúdos. Numa outra vertente, mais profissional, surgem os vídeos em tempo real, nomeadamente, em videovigilância. Nesta última, a estabilização tem vindo a ganhar um papel bastante importante pois as falhas causadas por erros ou instabilidades de um sistema podem ser cruciais e são cada vez menos aceitáveis.

A popularidade da estabilização de vídeo tem aumentado consideravelmente quando associada à ideia de uma fácil aquisição de vídeo em alta qualidade, mesmo em condições bastante adversas. Grande parte dos métodos de estabilização que já é possível encontrar em diversas câmaras não é digital. Estes métodos estabilizam as câmaras de um ponto de vista físico, conseguindo eliminar a maior parte dos movimentos indesejados, no entanto, tornam este tipo de câmaras demasiado caras pois são baseados em *hardware* muito sofisticado. Outros métodos não digitais podem tornar-se também inconvenientes para situações de videovigilância ou curtas filmagens amadoras, devido ao seu tamanho e peso excessivos.

Determinadas aplicações de estabilização de vídeo requerem soluções mais baratas, mais convenientes e igualmente eficazes. Assim, a estabilização de imagens enquadra-se neste projeto com o objetivo de tornar este tipo de processamento, que por si já é computacionalmente exigente, bastante mais rápido e eficaz.

Mas como poderá tal ser concebido? Nesta dissertação apresenta-se uma proposta que utiliza apenas a informação já capturada sem ser necessária a adição de qualquer tipo de sistema externo e sendo, ainda, mais eficiente que os sistemas já existentes.

A presente dissertação encontra-se dividida em quatro secções. No capítulo 3 apresentam-se todos os fundamentos teóricos que são base deste projeto. Começa-se por mostrar o estado de arte nesta área de desenvolvimento, passando-se depois aos conceitos

[1] http://hd-analytics.net/?q=overview - Técnico, I.S. and Observit. *High Definition Analytics Project*. 2011

teóricos inerentes à técnica de estabilização utilizada. No capítulo 4 é apresentada de forma pormenorizada toda a estratégia utilizada para a estabilização de vídeo, sendo explicado todo o processo de estimação de homografias aplicado. No capítulo 5 apresenta-se a técnica inovadora de utilização da compressão H.264 para melhorar significativamente a performance do estabilizador de vídeo implementado. No capítulo 6 é apresentada uma extensa análise que cobre os resultados obtidos na estabilização de um vídeo, bem como comparações importantes em termos de velocidade de processamento em cada caso estudado. Para finalizar, no capítulo 7 serão apresentadas todas as conclusões efetuadas e perspetivas futuras para este projeto.

3. FUNDAMENTOS TEÓRICOS

No capítulo que se segue descrevem-se os fundamentos teóricos que são a base deste projeto, assim como a preparação tomada em consideração antes da implementação do estabilizador de vídeo.

Atualmente, a indústria atingiu um ponto tal em que se torna difícil ir mais longe na forma como se constroem ou desenvolvem novos métodos de estabilização. Pensando nas vertentes de estabilização que recorrem a *hardware* torna-se cada vez mais difícil evoluir no sentido da construção de novos dispositivos físicos capazes de melhorar os efeitos pretendidos, deparando-se ainda com tempos e custos acrescidos na sua produção. Contrariamente, na era digital é possível ganhar muitos pontos a favor se se pensar numa ótica evolutiva de *software* e de novas tecnologias digitais que podem trazer inúmeros benefícios à estabilização digital de vídeo. É, portanto, necessário evoluir no sentido de uma tecnologia revolucionária ou, no caso aqui apresentado, recorrer às tecnologias mais avançadas, como a compressão de vídeo, tirando partido do que de melhor se pode obter delas. A compressão de vídeo apresenta-se, então, como uma área com muito espaço para evolução.

Primeiramente é importante saber o que é o processamento digital de imagem e perceber a diferença existente entre estabilização digital de imagem e estabilização digital de vídeo. Acerca destes dois últimos conceitos, o primeiro refere-se ao processo de correção de efeitos de movimentos indesejados durante a obtenção de uma imagem simples; o segundo diz respeito ao processo de eliminação de movimentos indesejados de uma câmara de vídeo durante uma sequência completa de imagens.

3.1. PROCESSAMENTO DIGITAL DE IMAGEM

O nascimento do computador introduziu na sociedade uma máquina muito potente no que respeita ao cálculo numérico computacional. Com a evolução ao longo dos anos apareceu a necessidade de juntar a esta potente máquina a capacidade de processamento de informação não-numérica, cedendo-lhe assim a possibilidade de decisão inteligente igual ou ainda mais eficiente que a do cérebro humano [3]. É nesta evolução, geralmente chamada de inteligência artificial, que o processamento digital de imagem se insere, abrangendo diferentes áreas de estudo. O processamento digital de imagem é uma técnica computacional de processamento de dados inserida na área de visão por computador. É o processo de receção e análise da informação visual executado por um computador. Porém, este processo não consiste apenas na extração de características, análise e reconhecimento de imagens, mas também na codificação, filtragem, melhoramento e restauro das mesmas [3].

Quando se fala em processamento de imagem, abrange-se um vasto conjunto de diferentes técnicas possíveis de aplicar sobre uma imagem (ou vídeo), modificando-a com finalidades diferentes. Entre estas distinguem-se, de uma forma geral, as seguintes:

- Processamento: melhoramento da qualidade de imagem através de filtragens e outros métodos;
- Discretização, representação e redução de tamanho: compressão codificada de bits para reduzir espaço de armazenamento;
- Análise (do inglês – *Image Analytics)*: extração de informação relevante, quantificação e segmentação de zonas, estabilização, etc. [3].

É, então, neste último contexto que surge a estabilização de imagem. Nos últimos anos, a estabilização tem vindo a crescer substancialmente sendo, geralmente, associada ao aumento de qualidade de vídeo que se obtém num dispositivo. Muitas formas de estabilização têm sido desenvolvidas até hoje, desde as estabilizações que recorrem somente a *hardware* até às puramente digitais que recorrem apenas a software e às informações extraídas das imagens analisadas.

3.2. TIPOS DE ESTABILIZAÇÃO

A variedade dos diferentes tipos de estabilização existentes no mercado começa a aumentar consideravelmente e apresenta atualmente pelo menos três vertentes, cada uma com as suas características, vantagens e desvantagens. Os diferentes tipos de estabilização são:

- Estabilização Mecânica
- Estabilização Ótica
- Estabilização Digital

Apesar do trabalho desenvolvido dizer respeito à estabilização digital de vídeo, devem conhecer-se as diferentes estabilizações existentes. Deste modo, estas estabilizações são descritas e comparadas neste capítulo, enfatizando-se a versão utilizada nesta dissertação.

3.2.1. ESTABILIZAÇÃO ÓTICA

Este é o tipo de estabilização mais comum e mais utilizada nas câmaras fotográficas. A estabilização ótica consiste num mecanismo capaz de estabilizar uma imagem antes de esta ser gravada no sensor CCD, obtendo-se assim uma qualidade muito superior nas imagens gravadas. Este mecanismo tem a capacidade de compensar os movimentos angular e translacional, geralmente conhecidos como *pan* e *tilt* das câmaras[2], consistindo num pequeno componente constituinte das lentes – o estabilizador – que se desloca no sentido inverso ao

[2] http://en.wikipedia.org/wiki/Image_stabilization - Encyclopedia, W.F. *Image Stabilization*

movimento da câmara durante o tempo de exposição numa fotografia ou filmagem. As primeiras lentes concebidas com a tecnologia de estabilização ótica foram introduzidas no mercado pela primeira vez em 1995 pela Canon[3]. Neste momento já começam a existir mais sistemas de estabilização, apesar de uma maneira geral, tratar-se de uma tecnologia bastante simples em que uma pequena lente interior compensa os movimentos indesejados da câmara. Porém, quando se está numa situação de deslocação da câmara ao longo de um plano é necessária a distinção entre os movimentos desejado e não desejado da câmara. Os novos sistemas são então constituídos ainda por um giroscópio capaz de medir as diferenças de velocidade em instantes diferentes, distinguindo assim estes dois tipos de movimento através da inexistência ou existência de aceleração.

Uma das primeiras tecnologias de estabilização ótica que surgiu é denominada de *Vari-Angle Prism Image Stabilizer* (VAP-IS) e pertence à Canon, sendo capaz de corrigir instabilidades em tempo-real com bastante eficácia. A Figura 3.1 ilustra o funcionamento deste sistema.

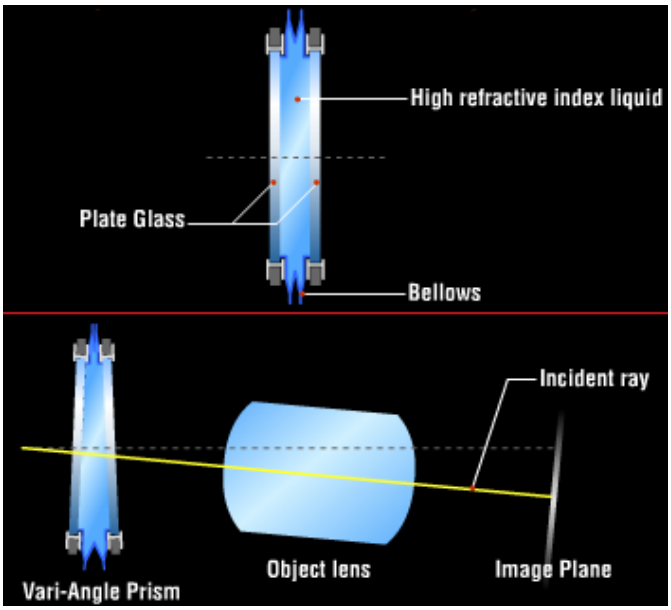


Figura 3.1 – Sistema *Vari-Angle Prism Image Stabilizer* [4].

Como se pode observar, esta tecnologia é baseada numa ótica flexível no interior de uma lente constituída por dois elementos de vidro liso separados por um líquido especial formando uma espécie de fole. Este pequeno fole expande ou contrai sempre que existe uma perturbação e o líquido, que possui um índice de refração bastante elevado, desvia o raio de luz na direção oposta, compensando desta forma os movimentos indesejados[4].

Outra tecnologia de estabilização ótica existente é a *Optical Shift Image Stabilizer* (OSIS), sendo esta a mais utilizada nas câmaras fotográficas de hoje em dia (Figura 3.2).

[3] http://web.canon.jp/imaging/lens/technology/index2.html - Inc, C. *Canon In-Lens Image Stabilizers*

[4] http://www.canon.com/bctv/faq/vari.html - Inc, C. *Vari-Angle Prisme Image Stabilizer (VAP-IS)*

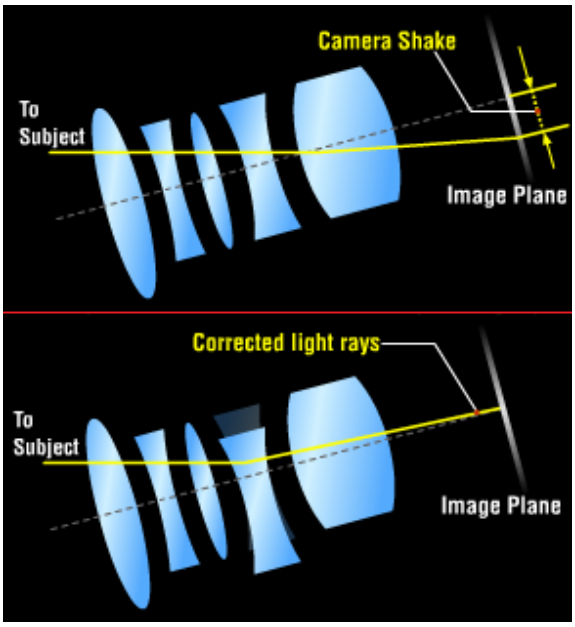


Figura 3.2 – Sistema *Optical Shift Image Stabilizer* [5].

A figura mostra que este sistema consiste num conjunto de lentes internas que se desloca paralelamente ao plano da imagem e perpendicular ao eixo ótico. Desta forma, os raios de luz que atingem o plano da imagem permanecem estáveis. Para além disto, existem ainda dois pequenos sensores (*shake-detecting sensors*) que detetam o ângulo e a velocidade do movimento e enviam essa informação para um microcontrolador de alta velocidade de 32 bits. Este, por sua vez, converte os sinais recebidos em movimentos inversos que atuam diretamente sobre a lente estabilizadora[5].

Apesar de este sistema ser bastante eficaz, apresenta a desvantagem de ser extremamente caro em comparação com os outros métodos de estabilização existentes, perdendo, por vezes, algum interesse. Em contrapartida, faz-se valer pelo seu desempenho ao compensar quase por completo qualquer tipo de vibração de uma câmara e preservando a qualidade de imagem desde o início até ao fim do processo de estabilização.

3.2.2. ESTABILIZAÇÃO MECÂNICA

Esta é uma tecnologia de estabilização um pouco diferente, pois passa por estabilizar toda a câmara como um inteiro e não apenas a imagem em si. De modo geral, este sistema é composto por um sensor que deteta deslocações da câmara, compensando desta forma o movimento indesejado da mesma. Assim, pode-se afirmar que neste tipo de estabilização é o próprio corpo da máquina que é estabilizado.

[5] http://www.canon.com/bctv/faq/optis.html - Inc, C. *Shift Optical Image Stabilizer (OSIS)*

Este tipo de estabilização utiliza um dispositivo denominado de "*Gyros*", que consiste num giroscópio constituído por duas pequenas rodas e uma bateria. A Figura 3.3 (a) que se apresenta em seguida mostra um sistema deste tipo.

(a) (b)

Figura 3.3 – Estabilizador mecânico: (a) *Gyros*; (b) estabilizador profissional.

Os giroscópios são sensores que, quando detetam movimento, enviam um sinal para motores ligados às pequenas rodas que fazem a câmara movimentar-se no sentido oposto à perturbação, mantendo-a assim estabilizado. Como se pode observar na mesma figura, os "*Gyros*" são dispositivos que se fixam à camara funcionando como um pequeno tripé invisível [4]. A Figura 3.3 (b) mostra um exemplo mais complexo de um estabilizador mecânico.

Os sistemas mecânicos de estabilização são considerados os mais eficazes nos resultados obtidos. No entanto, apresentam desvantagens cruciais na hora da sua escolha, tais como, o peso associado aos dispositivos que os compõem, o excessivo consumo de bateria provocado pelo uso dos motores que compensam os movimentos indesejados, e a necessidade de alguma experiência para o manuseamento destes aparelhos, Figura 3.3 (b). Para além disto, os motores que constituem estes sistemas exigem algoritmos de controlo bastante complexos de modo a serem capazes de compensar corretamente, e em tempo real, qualquer movimento.

3.2.3. ESTABILIZAÇÃO DIGITAL

A estabilização digital é o tipo de tecnologia que mais difere de todas as outras. Um sistema deste tipo não necessita de recorrer a qualquer tipo de *hardware*, ou seja, não há intervenção nenhuma por parte de dispositivos físicos que contribuem para o processo de estabilização, sendo que recorre somente a algoritmos desenvolvidos para esse efeito. É portanto, um sistema puramente dependente do *software*, pelo que tem ganho bastante interesse devido ao facto de que a evolução das tecnologias das câmaras amadoras e de vídeo vigilância, ou mesmo em telemóveis, não permitem o acoplamento de outros sistemas físicos a estes.

Este tipo de tecnologia é, por vezes, considerado menos adequada devido à deterioração que as imagens resultantes da estabilização podem apresentar no que diz respeito à sua resolução. No entanto, recorrendo a novas tecnologias de programação e outros processos digitais podem obter-se resultados extremamente positivos e tão bons ou melhores que os apresentados pelos outros tipos de estabilização. De uma forma geral, esta tecnologia estima o movimento indesejado da câmara comparando *frames* consecutivas e, de seguida, aplica uma determinada transformação à sequência de imagens do vídeo com o objetivo de compensar o movimento indesejado [5]. No que diz respeito aos métodos de estabilização, existe uma diversidade imensa, sendo que neste projeto se apresenta um diferente dos até agora desenvolvidos: utilização das características presentes na compressão de vídeo H.264.

Este é o tipo de estabilização que mais vertentes e diferentes processos apresenta. Quando se fala em estabilização digital é frequente dividir este processo em dois tipos: vertente 2D e 3D. Esta última vertente é muito menos explorada devido à sua complexidade a nível de algoritmos bem como de *hardware* necessário para processar estes algoritmos e obter os resultados pretendidos em tempo real. No entanto, pode apresentar resultados praticamente perfeitos. Por outro lado, existe a vertente 2D que tem vindo a ser bastante explorada nos últimos anos com o objetivo de encontrar as melhores técnicas capazes de produzir os melhores efeitos em tempo real, sem reduzir a qualidade das imagens originais.

Comparativamente com os outros métodos de estabilização o método digital apresenta uma clara desvantagem no que diz respeito à qualidade de vídeo final que se obtém deste processo. No entanto, a qualidade dos métodos já existentes tem vindo a melhorar significativamente e o processo aqui apresentado visa ter em conta esse fator, bem como o da velocidade de processamento, que costuma ser, igualmente, um ponto que desfavorece a estabilização digital. Por outro lado, as vantagens desta tecnologia residem, sobretudo, nos resultados quase perfeitos que se podem obter, dependendo, no entanto, em grande parte da complexidade dos algoritmos utilizados. Para além disto, os custos inerentes a este processo de estabilização são claramente mais reduzidos pois não existe a necessidade de recorrer a outro tipo de dispositivos físicos extremamente caros.

Evitar o movimento da câmara durante o tempo de exposição de um vídeo pode representar um método bastante eficaz para obter um vídeo estabilizado e visualmente agradável. Métodos digitais podem ser capazes de estimar o movimento indesejado de uma câmara mesmo depois do vídeo já ter sido capturado ou, em casos mais complexos e como seria desejável, durante o processo de captura no momento em que cada *frame* é adquirida. A isto chama-se processamento em tempo real (ou do inglês – *Real-Time Processing*).

O movimento nos vídeos pode ser causado por objetos que se deslocam ou por movimentos da câmara. A estabilização digital de imagem tem como objetivo compensar os movimentos indesejados da câmara, distinguindo aqueles que são os movimentos de objetos e que não devem ser corrigidos como sendo indesejados.

A base da estabilização digital de vídeo passa pela estimação dos movimentos da câmara através da análise das *frames* de um vídeo. O processo compara duas *frames* consecutivas de uma sequência de várias *frames*, estimando o movimento global (*Global Motion – GM*) entre ambas e compensando assim qualquer deslocamento indesejado que fora detetado [6]. Este processo é repetido até a sequência terminar.

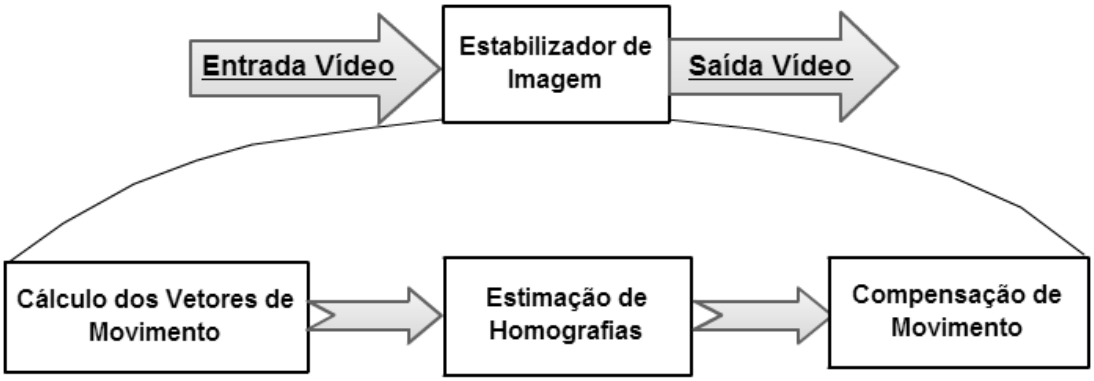


Figura 3.4 – Diagrama reduzido de um estabilizador de vídeo.

A Figura 3.4 mostra o referido sistema quando recebe um vídeo não estabilizado à entrada e devolve um vídeo estabilizado à saída. O primeiro passo na estabilização digital de vídeo passa pela estimação dos vetores de movimento em cada macro bloco de uma *frame*, o qual se denomina de *Local Motion Vectors* (LMV).

3.3. LOCAL MOTION ESTIMATION – DETEÇÃO DE KEYPOINTS

Cada *frame* de um vídeo pode ser dividida em vários blocos, formando os chamados Macro Blocos (MB). De uma maneira geral, cada MB de uma *frame* tem um tamanho definido de 16 por 16 pixéis, sendo que em alguns casos poder-se-ão ter MB divididos em duas ou quatro partes, obtendo-se, assim, alguns de 16x8, 8x16 ou 8x8. Eventualmente, poucos MB podem ainda ser divididos em duas ou quatro partes, obtendo-se agora MB de 8x4, 4x8 ou 4x4. Na Figura 3.5 entende-se melhor qual a definição de MB e como cada *frame* pode ser dividida para uma posterior análise.

Figura 3.5 – *Frame* dividida em MB de 16x16 pixéis.

A figura mostra uma *frame* de um vídeo dividida em vários MB de 16x16 pixéis. Quando comparadas duas *frames* consecutivas, de cada MB vai resultar um vetor de movimento denominado de *Motion Vector* (MV), que corresponde ao deslocamento desse MB ao longo do tempo de uma *frame* I_t para uma *frame* I_{t+1}. Porém, estimar um MV para cada MB pode tornar-se, computacionalmente, inexequível se se estiver perante um vídeo de alta qualidade, sendo assim impossível estabilizar um vídeo em tempo real. Desta forma, recorre-se a métodos que permitem calcular aquilo a que se denomina de *keypoints*. Um bom detetor de *keypoints* pode ser um dos pontos mais importantes de todo o processo de estabilização digital, pois é ele um dos fatores que pode tornar todo o processo significativamente mais rápido.

Local Motion Estimation é o processo de estimação do movimento de cada MB quando comparadas duas *frames* consecutivas, recorrendo ao cálculo dos *keypoints* através de um método de estimação de vetores de movimento. Os *keypoints* são os pontos correspondentes em cada *frame* I_t e I_{t+1} que melhor descrevem o movimento da imagem causado pela deslocação da câmara. Ou seja, nem todos os pontos representantes de cada MB são considerados *keypoints*, pelo que deverão ser devidamente escolhidos através de um método capaz de selecionar os melhores pontos tendo em conta a sua qualidade. Em processamento digital de imagem nenhuma imagem ou sequência de imagens pode ser analisada sem antes serem extraídas as suas melhores características. No caso em concreto aqui analisado pretende-se extrair os melhores pontos de uma *frame* e, como tal, o método denominado de *Good Features to Track* pode ser o melhor para obter os resultados mais satisfatórios neste processo.

O processo de *Local Motion Estimation* pode ser executado recorrendo a métodos de estimação de movimentos, através do cálculo de pontos de interesse em duas *frames* consecutivas. A conjugação de métodos como *Good Features to Track* e *Optical Flow* podem fornecer características essenciais para executar este processo.

3.3.1. GOOD FEATURES TO TRACK

Este é um dos métodos mais conhecidos para extrair as características de uma determinada imagem, fornecendo aquilo a que se denomina de pontos fortes ou *keypoints*. *Good Features to Track* é um método capaz de analisar as características de cada *pixel* de uma *frame*, devolvendo os melhores pontos, ou seja, os que apresentam as melhores características. Contudo, primeiro que tudo é necessário entender o verdadeiro significado de característica ou *feature* aqui descrita. Na verdade, não existe uma definição concreta para este termo e, geralmente, depende do problema que se quer analisar. Porém, *feature* pode ser definido como sendo uma parte interessante de uma imagem ou *frame* e que é utilizado como ponto de partida para o desenvolvimento de um algoritmo. Assim sendo, todo o algoritmo desenvolvido será tão bom quanto melhor for o método de deteção de *features* (*to track*) utilizado[6]. Existem vários métodos para deteção de *features* ou "deteção de cantos" (do inglês – *Corner Detection*), por isso, aqui apenas se apresenta aquele que foi utilizado, sendo descrito de forma sucinta.

Harris Corner Detection é um método (desenvolvido por *Harris* e *Stephens*), entre muitos outros, para deteção de "cantos" numa imagem ou sequência de imagens. O algoritmo de *Harris* e *Stephens* reside no princípio de que, num determinado ponto/*pixel* de uma *frame,* classificado como "canto" ou *keypoint*, a intensidade da imagem muda drasticamente em múltiplas direções. Ou seja, de uma forma mais simples, um ponto de uma *frame* é classificado como *keypoint* ou ponto de interesse se a sua intensidade mudar significativamente quando existe uma deslocação na sua zona envolvente [7]. O valor de comparação para a intensidade do ponto (*threshold value*) pode variar, sendo devolvidos mais ou menos pontos de interesse consoante o valor aplicado. A figura 3.6 mostra a deteção de *keypoints* numa *frame* de um vídeo em movimento.

Figura 3.6 – Representação de *keypoints* numa *frame.*

Como se pode observar pela figura, os *keypoints* correspondem, geralmente, aos contornos (*edges*) dos objetos presentes na imagem.

[6] http://en.wikipedia.org/wiki/Feature_detection_(computer_vision) - Wikipedia. *Feature Detection (computer vision)*

3.4. OPTICAL FLOW

A técnica de *Optical Flow* ou Fluxo Ótico é um dos métodos mais utilizados para extração dos vetores de movimento locais entre duas *frames*, isto é, o movimento de cada MB que melhor descreve o movimento global de uma *frame* para outra. Existem vários algoritmos para calcular o fluxo ótico de um determinado conjunto de *features*. Nesta dissertação apresenta-se o algoritmo desenvolvido por *Lukas* e *Kanade* utilizando o método com pirâmides. Neste passo da estabilização de vídeo, o objetivo desta técnica passa por obter os *keypoints* da *frame* corrente correspondentes aos *keypoints* da *frame* anterior.

Primeiramente é necessário entender o significado correto de fluxo ótico. Fluxo ótico é o padrão do movimento aparente de objetos, contornos ou superfícies de uma imagem, causados pelo movimento relativo entre um observador e um dispositivo de captura de imagem [8]. A Figura 3.7 mostra esse padrão de movimento representado por vetores desenhados a vermelho em cada MB correspondente a um *keypoint*.

Figura 3.7 – Vetores de movimento de uma *frame* extraídos pela técnica de *Optical Flow*.

O método *Lucas-Kanade* encontra a melhor correspondência entre duas imagens com um número de comparações muito mais reduzido do que outras técnicas, utilizando o método do cálculo do gradiente entre duas *frames* consecutivas de uma sequência de imagens. Esta técnica admite que o movimento é praticamente constante na vizinhança local de um pixel em estudo, resolvendo a equação do fluxo ótico para cada vizinho deste mesmo pixel [9].

Depois de extraídas as características necessárias para executar o processo de estabilização digital de imagem, é possível calcular a transformação a aplicar numa *frame,* de modo a compensar o movimento indesejado da câmara. Este é o processo mais complexo num algoritmo de estabilização de vídeo, pois é necessário começar por distinguir os movimentos

desejados e não desejados da câmara. Os movimentos considerados desejados num vídeo são todos aqueles que correspondem aos movimentos de objetos ao longo de uma sequência estática de *frames.* Contrariamente, os movimentos indesejados são todos aqueles que correspondem a deslocações aparentes de objetos, ou seja, quando existe uma deslocação da *frame* como um todo. Na Figura 3.8 (b) ilustra-se uma situação onde poderá existir movimento desejado que acontece quando há um objeto móvel (assinalado a vermelho) na *frame* (neste caso, as pessoas). Na Figura 3.8 (a) ilustra-se uma situação onde o movimento é sempre indesejado pois não existem objetos móveis na *frame*.

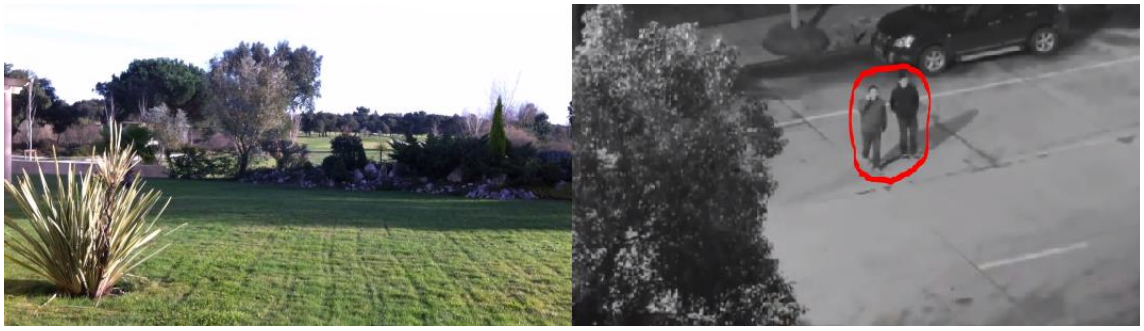


Figura 3.8 – Tipos de movimento: (a) movimento indesejado; (b) movimento desejado

Esta distinção é feita através da classificação dos *keypoints* em *inliers* e *outliers*. Um dos processos bastante eficazes para esta classificação é o algoritmo de RANSAC, um algoritmo baseado em probabilidades.

3.5. ALGORITMO RANSAC

RANSAC é a abreviação para o termo *RANdom SAmple Consensus* e corresponde a um algoritmo de estimação robusta de parâmetros de um modelo matemático constituído por um conjunto de dados. A ideia básica deste algoritmo pressupõe que um conjunto de dados a analisar é constituído por *inliers* e *outliers. Inliers* são todos os pontos do conjunto de dados que podem ser explicados através de um modelo de parâmetros; *outliers* são todos os pontos que fogem à regra, ou seja, são pontos que não obedecem ao mesmo modelo de parâmetros[7]. Observe-se a Figura 3.9 que ilustra uma aplicação simples de distinção entre *inliers* e *outliers*.

[7] http://wikipedia.qwika.com/en/RANSAC – Wikipedia, *RANSAC*

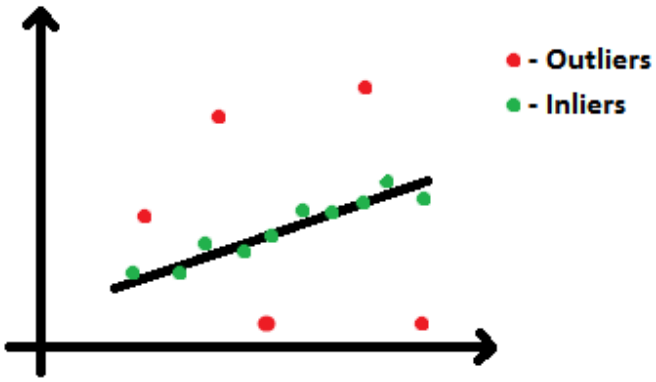


Figura 3.9 – Exemplo de uma classificação de *keypoints* como *inliers* e *outliers.*

Como se pode observar, todos os pontos que saem fora de um modelo simples (assinalado por uma linha a preto) são considerados *outliers*. É importante salientar que o facto de um ponto sair do modelo definido depende de um valor de *threshold* previamente escolhido, ou seja, existe um intervalo de erro que classifica os pontos como *inliers* e uma distância a partir da qual os pontos são considerados como *outliers*. Mas o que é um valor de *threshold*? É um valor definido empiricamente de forma que com uma probabilidade α o ponto em estudo é classificado como *inlier*, ou seja, apenas 5% dos pontos é incorretamente classificado como não sendo *inlier* [10]. Desta forma, assumindo para os cálculos um valor de $\alpha = 0{,}95$, o valor de *threshold* utilizado foi de 6,0 [10].

O algoritmo de RANSAC é capaz de interpretar/filtrar dados que contenham uma percentagem de erros significativa sendo, idealmente, recomendado para aplicações automatizadas de análise de imagens onde os dados analisados provêm de processos de deteção de *features* [11].

Numa aplicação mais concreta para a estabilização de vídeo, está-se perante um conjunto de pontos extraídos através de um detetor de *keypoints*. Entre os vários *keypoints*, alguns são considerados *inliers* quando se pretende calcular qual a homografia que transforma o conjunto de pontos de uma *frame* I_t no conjunto de pontos de uma *frame* I_{t+1}. Desta forma, a ideia passa por selecionar corretamente os *keypoints* considerados *inliers* e utilizá-los para o cálculo da homografia que melhor se adapta à situação em estudo. A isto chama-se de estimação robusta pois trata-se de uma estimação tolerante a *outliers* [10].

Existem várias homografias possíveis de estimar numa sequência de duas *frames*:

- Transformação Translacional
- Transformação Euclidiana
- Transformação Afim
- Transformação Projetiva

Dependendo do tipo de homografia que se pretende considerar, o algoritmo RANSAC pode ser mais ou menos complexo. No entanto, a ideia base deste algoritmo é bastante simples. Na transformação mais simples, a translacional por exemplo, a ideia passa pelo seguinte: dois

keypoints são selecionados aleatoriamente. Estes pontos formam uma linha, sendo o modelo atualmente em estudo. A importância atribuída a esta linha (que a classifica como sendo o modelo a seguir) é medido pelo número de pontos que está a uma determinada distância Euclidiana desta linha, não podendo exceder um determinado valor *threshold*. Todos os pontos que excedem este valor são considerados *outliers*, não contribuindo para aumentar o peso atribuído ao modelo. Este processo é repetido um certo número de vezes, sendo que o modelo a seguir é a linha com maior peso, isto é, com maior número de *inliers* [10]. No entanto, calcular o número de *inliers* para todos os modelos existentes pode tornar-se uma tarefa extremamente pesada computacionalmente. Desta forma, de acordo com este algoritmo, é possível definir um número de amostras N suficientemente grande para garantir que, com uma probabilidade p, no mínimo uma das amostras do conjunto de s pontos aleatórios é livre de *outliers*. Geralmente p é igual a 0,99; supondo que w é a probabilidade de um dos pontos escolhidos ser *inlier* e $\varepsilon = 1 - w$ a probabilidade de um dos pontos escolhidos ser *outlier*, então o número de amostras N pode ser calculado da seguinte forma:

$$(1 - w^s)^N = 1 - p \Leftrightarrow \boxed{N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)}} \qquad (3.1)$$

Em síntese, é desejável encontrar um modelo que descreva, através de parâmetros, um conjunto de pontos, recorrendo a um número de amostras simples suficientemente pequeno para determinar o modelo pretendido. A forma de implementação do algoritmo para cada homografia será explicada mais à frente no capítulo 4.

4. MÉTODO DE ESTABILIZAÇÃO IMPLEMENTADO

Neste capítulo, analisa-se todo o processo de estabilização utilizado neste projeto, sendo explicada detalhadamente cada parte do mesmo recorrendo a imagens de um vídeo amador capturado por uma câmara em HD (resolução = 1280x720) para mostrar os acontecimentos ao longo do vídeo.

O processo de desenvolvimento adotado para esta dissertação seguiu alguns requisitos já existentes até à data no projeto HDA. Assim, todo o algoritmo implementado foi desenvolvido em linguagem C++, de onde foi possível tirar partido das vantagens que esta linguagem traz em termos de velocidade bem como a utilização da biblioteca *OpenCV* de visão por computador. A biblioteca *OpenCV* é uma das mais utilizadas para o processamento digital de imagem a vários níveis. Contém inúmeros testes, funções e algoritmos implementados bastante úteis para a estabilização de imagens, como por exemplo os algoritmos de extração de *features* e técnicas de *Optical Flow*. É de salientar que foi utilizada a última versão desta biblioteca, 2.4.3.

A implementação do método de estabilização pode ser dividida em três partes: a primeira, respeitante ao cálculo dos vetores de movimento entre duas *frames* consecutivas, ao longo de um vídeo; a segunda, que procede à estimação da homografia pretendida na estabilização, sendo que foram implementadas quatros transformações diferentes por ordem de complexidade; finalmente, a última referente à compensação do movimento estimada através da homografia inversa, com o objetivo de estabilizar cada *frame* do vídeo. Para completar os objetivos desta tese de mestrado procedeu-se a um novo método de obtenção de vetores de movimento de forma a melhorar a velocidade de processamento do algoritmo de estabilização, no entanto, esta abordagem apenas será descrita no capítulo V.

Por forma a entender melhor o método de estabilização implementado, apresenta-se na Figura 4.1 um sistema mais detalhado em relação ao da Figura 3.4.

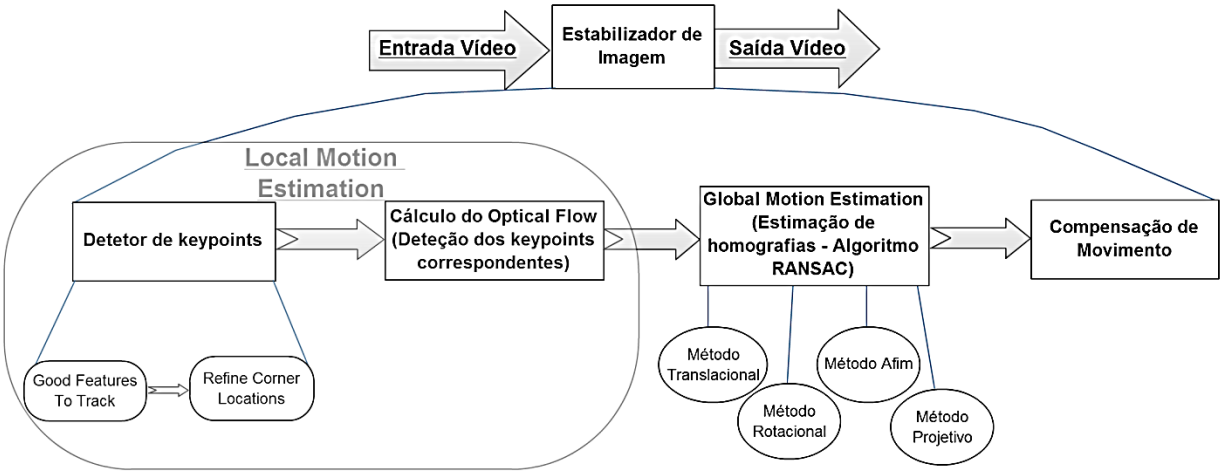


Figura 4.1 – Diagrama estendido de um estabilizador de vídeo.

Este sistema recebe à entrada um vídeo não estabilizado, processa a sua estabilização comparando *frames* consecutivamente e extrai um vídeo estabilizado. O processo de estabilização foi então implementado tomando em consideração os fundamentos teóricos descritos no capítulo anterior.

4.1. DETETOR DE KEYPOINTS E REFINAMENTO DAS SUAS POSIÇÕES

O primeiro passo implementado na estabilização de vídeo passa pela deteção dos *keypoints* de uma *frame* I_t. Este ponto permite calcular o movimento de cada MB com interesse significativo. Desta forma, procedeu-se à deteção de *features* numa imagem, recorrendo ao algoritmo *Good Features to Track*. Através deste método obtiveram-se todos os pontos de interesse ou *keypoints* da *frame* em análise. Por fim, para tornar mais preciso o cálculo destes pontos, procedeu-se ao refinamento da posição dos mesmos através de um método fornecido pela biblioteca *OpenCV*.

4.2. TÉCNICA OPTICAL FLOW

Para se obter o movimento indesejado de uma *frame* e depois corrigi-lo é necessário calcular os vetores de movimento em cada MB com interesse, ou seja, é necessário obter os *keypoints* da *frame* I_{t+1} correspondentes aos já encontrados no primeiro passo. Assim, procedeu-se ao cálculo do fluxo ótico da *frame* I_t para a *frame* I_{t+1}, recorrendo ao algoritmo de *Lucas-Kanade* descrito no capítulo anterior.

4.3. TRANSFORMAÇÕES SOBRE UMA IMAGEM

Terminado o segundo passo têm-se dois conjuntos de pontos, um por cada uma das duas *frames* em análise, sendo então possível calcular o movimento global de uma *frame* através da estimação da homografia que transforma os *keypoints* da *frame* I_t nos *keypoints* da *frame* I_{t+1}. Como mencionado no capítulo III, existem quatro tipos de transformações possíveis de estimar, translacional, euclidiana, afim e projetiva, cujas implementações foram executadas na íntegra e são explicadas em seguida. Para começar, explica-se em detalhe como é feita a base da estimação de qualquer transformação ou homografia.

4.3.1. ESTIMAÇÃO DE UMA HOMOGRAFIA

Dados dois conjuntos P e P' de pontos de duas *frames* consecutivas é possível estimar a homografia que transforma o primeiro conjunto no segundo.

$$P = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$
$$P' = \{(x'_1, y'_1), (x'_2, y'_2), \dots, (x'_n, y'_n)\}$$

A ideia passa pela estimação de uma homografia definida por uma matriz 3x3, H, que descreve a transformação dos pontos x_i nos pontos x'_i, ou vice-versa, de tal forma que:

$$x'_i = H \cdot x_i \qquad \text{ou} \qquad x_i = H^{-1} \cdot x'_i \qquad (4.1) \text{ e } (4.2)$$

Uma vez que H é uma matriz de tamanho 3x3, é necessário que as equações (4.1) e (4.2) envolvam vetores homogéneos. Ou seja, por definição, um ponto (X, Y) num sistema de coordenadas cartesianas é representado, em coordenadas homogéneas, por $[x, y, w]$ onde $X = \frac{x}{w}$ e $Y = \frac{y}{w}$. É de salientar que um ponto homogéneo a duas dimensões não terá três "graus de liberdade" (*degrees of freedom*), mas apenas dois pois o valor de *w* é arbitrário e para o caso em estudo será sempre 1 [10]. Assim, daqui em diante um ponto homogéneo será designado, na sua forma matricial, como:

$$P_i = [x_i, y_i, 1]^T = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \qquad (4.3)$$

Dependendo da transformação que se pretende estimar, existem mais ou menos parâmetros resultantes dessa estimação. Assim, numa transformação translacional estimam-se 2 parâmetros; numa euclidiana estimam-se 3 parâmetros; numa afim estimam-se 7 parâmetros; e, finalmente, numa projetiva estimam-se 9 parâmetros. Estes parâmetros serão explicados nos respetivos subcapítulos.

A primeira questão que se coloca neste processo de obtenção de homografias é quantas correspondências $x_i \rightarrow x'_i$ são necessárias para se calcular uma transformação. Obviamente que a resposta dependerá do número de parâmetros que se pretendem aferir, ou seja, depende da transformação que se quer calcular numa imagem. No entanto, antes de se iniciar a obtenção das homografias é importante perceber a importância de uma normalização dos dados que serão base dos cálculos a efetuar.

4.3.2. NORMALIZAÇÃO

A normalização de dados é um processo bastante importante quando se toma em consideração uma transformação mais complexa por isso, a importância deste passo é sobretudo aplicável na última transformação. Recorrendo a um exemplo mais concreto, sem normalização os pontos x_i , x'_i de uma imagem são de uma ordem de grandeza do tipo

$[x_i, y_i, w]^T = [100,100,1]^T$, ou seja, x, y têm um valor muito maior que w. Como se verá mais à frente, alguns métodos utilizados para calcular as homografias podem apresentar valores que, nesta ordem de grandeza, poderão atingir, em determinados cálculos, 10^4. Estes são valores bastante grandes quando se pretende manipulá-los em conjunto com outros muito mais pequenos, podendo trazer erros associados aos cálculos matemáticos envolvidos. Poder-se-ia dizer que na presença de um conjunto de dados exatos e uma precisão infinita nos mesmos, a normalização seria dispensável. Porém, em grande parte dos casos (senão na sua totalidade), existe sempre a presença de ruído o que poderá fazer divergir muito o resultado final. A conclusão aqui a tirar é que a normalização dos dados proporciona resultados extremamente mais eficientes para a estabilização de vídeo e, por isso, não deverá ser considerada uma opção [10]. Por conseguinte, a normalização aplicada neste trabalho passa por efetuar uma transformação de similaridade sobre os pontos x_i e x_i', ou seja, uma transformação que consiste numa translação e escalamento dos pontos constituintes do conjunto de dados, para que os pontos x_i se transformem nos pontos $\tilde{x}_i$ e os pontos x_i' se transformem nos pontos $\tilde{x}'_i$. Desta forma, o centróide dos pontos $\tilde{x}_i$ ou $\tilde{x}'_i$ passa a ser o ponto $(0,0)$ e a distância média à origem é $\sqrt{2}$ [10].

A normalização apresenta-se em seguida sob a forma matricial. Dados os pontos

$$p_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad \text{e} \quad p'_i = \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix}$$

e respetivas normalizações N e N', resultam os pontos normalizados

$$\tilde{p}_i = \begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ 1 \end{bmatrix} \quad \text{e} \quad \tilde{p}'_i = \begin{bmatrix} \tilde{x}'_i \\ \tilde{y}'_i \\ 1 \end{bmatrix}.$$

Ora a normalização é feita para cada ponto

$$\tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x},\ \tilde{y}_i = \frac{y_i - \mu_y}{\sigma_y} \quad \text{e} \quad \tilde{x}'_i = \frac{x'_i - \mu_{x'}}{\sigma_{x'}},\ \tilde{y}'_i = \frac{y'_i - \mu_{y'}}{\sigma_{y'}} \qquad (4.4) \text{ e } (4.5)$$

Então, aplicando o mesmo para os conjuntos de pontos P e P', obtêm-se os conjuntos $\tilde{P}$ e $\widetilde{P'}$ resultantes da aplicação das matrizes de normalização

$$N = \begin{bmatrix} \frac{1}{\sigma_x} & 0 & -\frac{\mu_x}{\sigma_x} \\ 0 & \frac{1}{\sigma_y} & -\frac{\mu_y}{\sigma_y} \\ 0 & 0 & 1 \end{bmatrix} \quad \text{e} \quad N' = \begin{bmatrix} \frac{1}{\sigma_{x'}} & 0 & -\frac{\mu_{x'}}{\sigma_{x'}} \\ 0 & \frac{1}{\sigma_{y'}} & -\frac{\mu_{y'}}{\sigma_{y'}} \\ 0 & 0 & 1 \end{bmatrix}$$

Por fim, a transformação a obter depois da normalização dos pontos será a vertente normalizada, $\tilde{T}$, pelo que deverá ser desnormalizada para se ter os verdadeiros valores a aplicar sobre a imagem. Assim, a matriz que traduz a transformação a aplicar sobre uma imagem é calculada através da desnormalização.

Sabe-se que

$$\widetilde{P'} = \tilde{T} \cdot \tilde{P}$$

Então a transformação T é dada por:

$$\Longrightarrow N' \cdot P_i' = \tilde{T} \cdot N \cdot P_i \Leftrightarrow P_i' = \underbrace{N'^{-1} \cdot \tilde{T} \cdot N}_{T} \cdot P_i \Leftrightarrow P_i' = T \cdot P_i$$

$$\Longrightarrow N' \cdot P_i' = \tilde{T} \cdot N \cdot P_i \Leftrightarrow \boxed{T = N'^{-1} \cdot \tilde{T} \cdot N} \qquad (4.6)$$

Depois de entendida a forma como se calcula uma homografia entre conjuntos de pontos de uma *frame*, explica-se agora o processo para obtenção dos parâmetros de cada uma das transformações aplicadas neste projeto.

É neste ponto da estabilização que o algoritmo RANSAC tem o papel mais importante. Encontrados os conjuntos de pontos em cada *frame* I_t e I_{t+1} é necessário averiguar aqueles que entram para o cálculo da transformação, ou seja, é preciso distinguir quais os *inliers* e os *outliers* nos conjuntos dados. Este processo distingue-se em cada transformação pelo número de pontos que fazem parte de um modelo a seguir (definido no algoritmo RANSAC), dependendo do número de parâmetros a calcular.

4.3.3. TRANSFORMAÇÃO TRANSLACIONAL

Esta é a transformação mais simples e possível de aplicar sobre cada *frame*. Ela corrige apenas movimentos indesejados horizontais e verticais. Para uma transformação translacional, é preciso 1 ponto de cada uma das *frames* (I_t e I_{t+1}) para servir de modelo, pois o número de parâmetros a obter é 2 (translação em x e em y). Assim, aleatoriamente será escolhido 1 ponto num conjunto de *keypoints* e o seu correspondente no outro conjunto, sendo executado o sistema de equações, para cada ponto, que define a translação de uma imagem.

$$\begin{cases} x_i' = x_i + T_x \\ y_i' = y_i + T_y \end{cases} \qquad (4.7)$$

que em forma matricial se representa por:

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \qquad (4.8)$$

Para se estimar a transformação translacional a aplicar em cada ponto começa-se por escolher 1 ponto de um conjunto de *keypoints* e o seu correspondente no outro, calculando-se depois o sistema matricial.

- Primeiro passo:

  O modelo é definido calculando-se o sistema matricial definido em cima para os dois pontos (um *match* de *keypoints*).

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad (4.9)$$

Da resolução deste sistema surgem os parâmetros horizontal e vertical que servem de modelo definido pelo algoritmo RANSAC. A obtenção destes parâmetros pode ser facilmente simplificada calculando-se apenas a diferença entre os dois pontos.

- Segundo passo:

  Depois de definido o modelo calculam-se as hipóteses definidas pelos outros *keypoints*, ou seja, a diferença entre os pontos correspondentes.
  Para cada *keypoint* calcula-se

$$\begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \qquad (4.10)$$

  sendo $\tilde{P}(\tilde{x}, \tilde{y})$ o valor do ponto estimado

- Terceiro passo:

  Comparam-se as hipóteses com o modelo definido. Todas as hipóteses que se encontrarem dentro de um determinado intervalo definido pelo valor de *threshold* são classificadas como *inlier*. Contrariamente, as que se encontrarem fora desse intervalo são classificadas como *outliers* e não são contabilizadas para estimar a transformação translacional.

$$Resultado = Modelo^2 - Hipótese^2$$
$$Resultado = (x_M^2 + y_M^2) - (x_H^2 + y_H^2)$$

  Numa situação ideal de um *inlier* ter-se-ia o valor do $Resultado$ em 0 (zero), o que significaria que a hipótese em estudo seria igual ao modelo proposto. Porém, devido ao ruído associado às *frames,* ou caso o ponto seja parte de um objeto em movimento, este valor pode oscilar e por isso, considera-se que existe um intervalo definido pelo *threshold* que distingue os *inliers* dos *outliers*.

$$\text{Se} \quad \begin{cases} Resultado < Threshold \\ Resultado > Threshold' \end{cases} \quad \text{então} \quad \begin{cases} Ponto\ \text{é}\ inlier \\ Ponto\ \text{é}\ outlier \end{cases}$$

O segundo e terceiro passos são repetidos o número de vezes necessárias até se encontrar o modelo que seja constituído pelo maior número de *inliers* e, assim, que melhor defina a transformação a estimar. De acordo com o algoritmo RANSAC, o número de vezes a executar este passo pode ser estimado calculando a fórmula já definida e, assumindo que a percentagem de outliers nas *frames* pode rondar os 50%:

$$N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} \qquad (4.11)$$

$$N = \frac{\log(1 - 0{,}99)}{\log(1 - (1 - 0{,}5)^1)}$$

$$N \simeq 7 \text{ vezes}$$

- Quarto passo:

  Finalmente, com os *inliers* todos encontrados pode-se estimar corretamente a transformação da *frame* I_t para a I_{t+1}. Neste caso, a estimação pode ser bastante simplificada, bastando calcular a média aritmética entre as distâncias calculadas no terceiro passo para o modelo definido. Assim, a transformação translacional a aplicar sobre cada *frame* em análise será:

$$H_T = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix},$$

  sendo T_x o valor da translação horizontal e T_y o valor da translação vertical.

  Um exemplo de aplicação prática desta transformação no projeto apresenta-se na sequência de 5 *frames* consecutivas na Figura 4.2 (não estabilizadas em cima (a) e estabilizadas em baixo (b)).

Figura 4.2 – Sequência de 5 *frames* consecutivas ilustrando o movimento translacional: (a) sequência não estabilizada – em cima; (b) sequência estabilizada – em baixo.

4.3.4. TRANSFORMAÇÃO EUCLIDIANA

A transformação euclidiana é um pouco mais complexa do que a anterior. Ela corrige, para além dos movimentos translacionais, os movimentos rotacionais indesejados. Para esta transformação são precisos 2 pontos de cada uma das *frames* (I_t e I_{t+1}) para servir de modelo, pois o número de parâmetros a obter é 3 (translação em x e em y e, ainda, rotação θ). Assim, aleatoriamente serão escolhidos 2 pontos de cada conjunto de *keypoints*, sendo executado o sistema de equações, para cada ponto, que define a rotação e translação de uma imagem:

$$\begin{cases} x_i' = x_i \cdot \cos\theta - y_i \cdot \sin\theta + T_x \\ y_i' = x_i \cdot \sin\theta + y_i \cdot \cos\theta + T_y \end{cases} \quad (4.12)$$

que em forma matricial se representa por:

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & T_x \\ \sin\theta & \cos\theta & T_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad (4.13)$$

Os passos a seguir para a estimação da transformação euclidiana apresentam algumas diferenças em relação àqueles seguidos para a translacional, começando agora por se escolher, aleatoriamente, 2 pontos num conjunto de *keypoints* e os seus correspondentes no outro conjunto, sendo calculado o sistema matricial com os dois pontos.

- Primeiro passo:

  O modelo é mais uma vez definido calculando-se agora o sistema matricial representado em cima para quatro pontos (dois *matches* de *keypoints*), o equivalente a um sistema de quatro equações para quatro incógnitas.

$$\begin{bmatrix} x_i' \\ y_i' \\ x_j' \\ y_j' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & T_x \\ \sin\theta & \cos\theta & T_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ x_j \\ y_j \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} x_i' \\ y_i' \\ x_j' \\ y_j' \end{bmatrix} = \begin{bmatrix} a & -b & T_x \\ b & a & T_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ x_j \\ y_j \end{bmatrix} \quad (4.14)$$

  sendo $a = \cos\theta$ e $b = \sin\theta$

  A ideia passa pela estimação da matriz 3x3 que transforma os pontos i e j nos pontos i' e j'. A melhor solução para este sistema de equações está na sua resolução através de um algoritmo denominado de *Direct Linear Transformation* (DLT). Assim, têm-se quatro equações

$$\begin{cases} x_i' = x_i \cdot \cos\theta - y_i \cdot \sin\theta + T_x \\ y_i' = x_i \cdot \sin\theta + y_i \cdot \cos\theta + T_y \\ x_j' = x_j \cdot \cos\theta - y_j \cdot \sin\theta + T_x \\ y_j' = x_j \cdot \sin\theta + y_j \cdot \cos\theta + T_y \end{cases} \quad (4.15)$$

  que através do algoritmo DLT se transformam no sistema matricial

$$\begin{bmatrix} x_i & -y_i & 1 & 0 \\ y_i & x_i & 0 & 1 \\ x_j & -y_j & 1 & 0 \\ y_j & x_j & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ T_x \\ T_y \end{bmatrix} = \begin{bmatrix} x'_i \\ y'_i \\ x'_j \\ y'_j \end{bmatrix} \quad (4.16)$$

  Da resolução deste sistema extraem-se os parâmetros que definem o modelo a seguir.

- Segundo passo:

Depois de definido o modelo aplica-se a cada *keypoint* da *frame* I_t uma matriz transformação definida pelos parâmetros do modelo, obtendo-se um novo ponto designado hipótese. Este ponto é subtraído ao seu real correspondente calculando-se, assim, a distância euclidiana entre ambos.

Para cada *keypoint* calcula-se

$$\begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ 1 \end{bmatrix} = \begin{bmatrix} a & -b & T_x \\ b & a & T_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \qquad (4.17)$$

sendo $\tilde{P}(\tilde{x}, \tilde{y})$ o valor do ponto estimado.

- Terceiro passo:

Depois de obtido este ponto, comparam-se todas as hipóteses obtidas, isto é, todos os pontos estimados $\tilde{P}(\tilde{x}, \tilde{y})$, sendo calculada a distância euclidiana ao seu real ponto correspondente $P'(x', y')$.

$$Resultado = \tilde{P}^2 - P'^2$$

$$Resultado = (\tilde{x}^2 + \tilde{y}^2) - (x'^2 + y'^2)$$

Numa situação ideal de um *inlier* seria de esperar que este valor fosse 0 (zero), porém, tal como nas outras transformações, pode existir ruído ou o ponto ser parte de um objeto em movimento, sendo classificado como outlier. Assim tem-se:

$$\text{Se} \quad \begin{cases} Resultado < Threshold \\ Resultado > Threshold \text{'} \end{cases} \quad \text{então} \quad \begin{cases} Ponto \text{ é } inlier \\ Ponto \text{ é } outlier \end{cases}$$

Os passos 2 e 3 são repetidos o número de vezes necessárias até ser encontrado o melhor modelo que define a transformação entre as duas *frames* consecutivas I_t e I_{t+1}, ou seja, aquele que apresenta o maior número de *inliers*. Desta forma, segundo o algoritmo RANSAC, este número pode ser significativamente reduzido e com fiabilidade bastante alta através da equação (4.11):

$$N = \frac{\log(1 - 0{,}99)}{\log(1 - (1 - 0{,}5)^2)}$$

$$N \simeq 17 \text{ vezes}$$

- Quarto passo

Finalmente, com os *inliers* todos encontrados é possível estimar a transformação entre as duas *frames*. Neste caso a transformação é estimada de forma diferente, sendo calculada, mais uma vez, através do algoritmo DLT, mas agora para um sistema de

equações sobredimensionado pois incluem-se todos os *inliers* encontrados. Assim, tem-se o sistema de n equações:

$$\begin{cases} x'_1 = x_1 \cdot \cos\theta - y_1 \cdot \sin\theta + T_{x_1} \\ y'_1 = x_1 \cdot \sin\theta + y_1 \cdot \cos\theta + T_{y_1} \\ \vdots \\ x'_n = x_n \cdot \cos\theta - y_n \cdot \sin\theta + T_{x_n} \\ y'_n = x_n \cdot \sin\theta + y_n \cdot \cos\theta + T_{y_n} \end{cases} \quad (4.19)$$

que através do algoritmo DLT se transformam no sistema matricial

$$\begin{bmatrix} x_1 & -y_1 & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ x_2 & -y_2 & 1 & 0 \\ y_2 & x_2 & 0 & 1 \\ & \vdots & & \\ x_n & -y_n & 1 & 0 \\ y_n & x_n & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ T_x \\ T_y \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix} \quad (4.20)$$

Deste sistema matricial resultam os parâmetros que vão definir a matriz de transformação euclidiana H_R, definida assim

$$\boxed{H_R = \begin{bmatrix} a & -b & T_x \\ b & a & T_y \\ 0 & 0 & 1 \end{bmatrix}} \quad (4.21)$$

4.3.5. TRANSFORMAÇÃO AFIM

Esta transformação apresenta um nível de complexidade maior devido ao número de parâmetros que se podem estabilizar numa imagem. Para além daqueles já vistos na transformação euclidiana, ela corrige escalamentos horizontais e verticais, compressão e *shear* (estes dois últimos parâmetros são responsáveis, sobretudo, pela deformação que ocorre nos objetos em movimentos mais bruscos). Na estimação desta transformação são necessários, no mínimo, 3 pontos de cada uma das duas *frames* consecutivas em análise, pois o número de parâmetros agora a estimar é de 7.

Assim, aleatoriamente serão escolhidos 3 pontos de cada conjunto de *keypoints*, sendo calculado, para cada ponto, o sistema de equações que se segue:

$$\begin{cases} x'_i = x_i \cdot a_{11} + y_i \cdot a_{12} + T_x \\ y'_i = x_i \cdot a_{21} + y_i \cdot a_{22} + T_y \end{cases} \quad (4.22)$$

que em forma matricial se representa por:

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & T_x \\ a_{21} & a_{22} & T_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad (4.23)$$

Os passos a seguir para a estimação desta transformação são idênticos aos utilizados para a transformação euclidiana, com a diferença da utilização de mais um ponto para o efeito. Começa-se então por escolher, aleatoriamente, 3 pontos num conjunto de *keypoints* e os seus correspondentes no outro conjunto, sendo calculado o sistema matricial com os três pontos.

- Primeiro passo:

  O modelo é mais uma vez definido através do cálculo do sistema matricial representado em cima para seis pontos (três *matches* de *keypoints*), o equivalente a um sistema de seis equações para seis incógnitas (os seis parâmetros mencionados dizem respeito às incógnitas a_{11}, a_{12}, a_{21}, a_{22}, T_x e T_y, no entanto, os 7 parâmetros serão deduzidos a partir destes como será apresentado em diante).

$$\begin{bmatrix} x_i' \\ y_i' \\ x_j' \\ y_j' \\ x_k' \\ y_k' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & T_x \\ a_{21} & a_{22} & T_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ x_j \\ y_j \\ x_k \\ y_k \end{bmatrix} \qquad (4.24)$$

  sendo

  $a_{11} = \cos\theta \cdot \lambda_1 \cdot \zeta$,

  $a_{12} = \cos\theta \cdot \lambda_1 \cdot \alpha - \frac{\lambda_2 \cdot \sin\theta}{\zeta}$,

  $a_{21} = \sin\theta \cdot \lambda_1 \cdot \zeta$, e

  $a_{22} = \sin\theta \cdot \lambda_1 \cdot \alpha - \frac{\lambda_2 \cdot \cos\theta}{\zeta}$.

  Estes parâmetros da matriz são definidos desta forma, pois a matriz de transformação afim pode ser decomposta na multiplicação da matriz euclidiana, de escalamento, de compressão e *shear*, como se apresenta de seguida:

$$A = \begin{bmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{bmatrix} = H_R \cdot H_S \cdot H_D = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \cdot \begin{bmatrix} \zeta & \alpha \\ 0 & \frac{1}{\zeta} \end{bmatrix} \qquad (4.25)$$

  A ideia passa, mais uma vez, pela estimação da matriz 3x3 que transforma agora os pontos i, j e k nos pontos i', j' e k' e, por fim, na estimação de cada uma das matrizes que a decompõem. A melhor solução para este sistema de equações está na sua resolução utilizando outra vez o algoritmo DLT. Assim, têm-se seis equações

$$\begin{cases} x_i' = x_i \cdot a_{11} + y_i \cdot a_{12} + T_x \\ y_i' = x_i \cdot a_{21} + y_i \cdot a_{22} + T_y \\ x_j' = x_j \cdot a_{11} + y_j \cdot a_{12} + T_x \\ y_j' = x_j \cdot a_{21} + y_j \cdot a_{22} + T_y \\ x_k' = x_k \cdot a_{11} + y_k \cdot a_{12} + T_x \\ y_k' = x_k \cdot a_{21} + y_k \cdot a_{22} + T_y \end{cases} \qquad (4.26)$$

que através do algoritmo DLT se transformam no sistema matricial

$$\begin{bmatrix} x_i & y_i & 0 & 0 & 1 & 0 \\ y_i & x_i & 0 & 0 & 0 & 1 \\ x_j & y_j & 0 & 0 & 1 & 0 \\ y_j & x_j & 0 & 0 & 0 & 1 \\ x_k & y_k & 0 & 0 & 1 & 0 \\ y_k & x_k & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \\ T_x \\ T_y \end{bmatrix} = \begin{bmatrix} x'_i \\ y'_i \\ x'_j \\ y'_j \\ x'_k \\ y'_k \end{bmatrix} \qquad (4.27)$$

Da resolução deste sistema extraem-se os parâmetros que definem o modelo a seguir. Para se obter cada um dos 7 parâmetros mencionados para este modelo, procedem-se a alguns cálculos matriciais e outros matemáticos.

- Rotação:

$$\frac{a_{11}}{a_{21}} = \frac{\cos\theta \cdot \lambda_1 \cdot \zeta}{\sin\theta \cdot \lambda_1 \cdot \zeta} = \frac{\cos\theta}{\sin\theta} = \tan\theta \Rightarrow \arctan\frac{a_{11}}{a_{22}} = \theta$$

$$H_R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \qquad (4.28) \text{ e } (4.29)$$

- Escalamento horizontal e vertical:

  Para estimar os dois parâmetros de escalamento da imagem procede-se ao cálculo matricial designado por *Singular Value Decomposition* (SVD), ou seja, Decomposição em Valores Singulares.

  A partir do algoritmo SVD aplicado à matriz afim A é possível extrair os seus valores singulares que são os parâmetros dos escalamentos, λ_1 e λ_2.

$$SVD(A) = SVD\left(\begin{bmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{bmatrix}\right) \Rightarrow H_S = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \qquad (4.30)$$

- Compressão e *Shear*:

  Manipulando os valores a_{11}, a_{12}, a_{21} e a_{22} é possível extrair os restantes parâmetros.

$$\frac{\sqrt{a_{11} \cdot a_{21}}}{\lambda_1} = \frac{\sqrt{(\cos\theta \cdot \lambda_1 \cdot \zeta)(\sin\theta \cdot \lambda_1 \cdot \zeta)}}{\lambda_1} \Rightarrow \zeta \qquad (4.31)$$

$$\frac{(a_{11} \cdot a_{12} + a_{21} \cdot a_{22})}{\zeta \cdot \lambda_1^2}$$

$$= \frac{(\cos\theta \cdot \lambda_1 \cdot \zeta)\left(\cos\theta \cdot \lambda_1 \cdot \alpha - \frac{\lambda_2 \cdot \sin\theta}{\zeta}\right) + \left(\sin\theta \cdot \lambda_1 \cdot \alpha - \frac{\lambda_2 \cdot \cos\theta}{\zeta}\right)(\sin\theta \cdot \lambda_1 \cdot \zeta)}{\zeta \cdot \lambda_1^2} \Rightarrow \alpha$$

$$H_D = \begin{bmatrix} \zeta & \alpha \\ 0 & \frac{1}{\zeta} \end{bmatrix} \qquad (4.32) \text{ e } (4.33)$$

- Segundo passo:

  O segundo passo é idêntico ao descrito na transformação euclidiana. Depois de definido o modelo aplica-se a cada *keypoint* da *frame* I_t uma matriz transformação definida pelos parâmetros do modelo, obtendo-se um novo ponto designado hipótese.

Este ponto é subtraído ao seu real correspondente calculando-se, assim, a distância euclidiana entre ambos.

Para cada *keypoint* calcula-se

$$\begin{bmatrix}\tilde{x}_i\\ \tilde{y}_i\\ 1\end{bmatrix} = \begin{bmatrix}a_{11} & a_{12} & T_x\\ a_{21} & a_{22} & T_y\\ 0 & 0 & 1\end{bmatrix} \cdot \begin{bmatrix}x_i\\ y_i\\ 1\end{bmatrix} \qquad (4.34)$$

sendo $\tilde{P}(\tilde{x}, \tilde{y})$ o valor do ponto estimado.

- Terceiro passo:

  Depois de obtido este ponto, comparam-se todas as hipóteses obtidas, isto é, todos os pontos estimados $\tilde{P}(\tilde{x}, \tilde{y})$, sendo calculada a distância euclidiana ao seu real ponto correspondente $P'(x', y')$.

$$Resultado = \tilde{P}^2 - P'^2$$

$$Resultado = (\tilde{x}^2 + \tilde{y}^2) - (x'^2 + y'^2)$$

Numa situação ideal de um *inlier* seria de esperar que este valor fosse 0 (zero), porém, tal como nas outras transformações pode existir ruído ou o ponto ser parte de um objeto em movimento, sendo classificado como outlier. Assim, tem-se:

$$\text{Se } \begin{cases} Resultado < Threshold \\ Resultado > Threshold' \end{cases} \text{ então } \begin{cases} Ponto \text{ é } inlier \\ Ponto \text{ é } outlier \end{cases}$$

Tal como nas transformações anteriores, os passos 2 e 3 são repetidos até ser encontrado o melhor modelo que define a transformação entre as duas *frames* consecutivas I_t e I_{t+1}, ou seja, aquele que apresenta o maior número de *inliers*. Desta forma, segundo o algoritmo RANSAC, este número pode ser significativamente reduzido e com fiabilidade bastante alta através da equação (4.11):

$$N = \frac{\log(1 - 0{,}99)}{\log(1 - (1 - 0{,}5)^3)}$$

$$N \simeq 35 \text{ vezes}$$

- Quarto passo

  Finalmente, com os *inliers* todos encontrados é possível estimar a transformação entre as duas *frames*. Mais uma vez, através do algoritmo DLT, mas agora para um sistema de equações sobredimensionado pois incluem-se todos os *inliers* encontrados, tem-se o sistema de n equações:

$$\begin{cases} x'_1 = x_1 \cdot a_{11} + y_1 \cdot a_{12} + T_{x_1} \\ y'_1 = x_1 \cdot a_{21} + y_1 \cdot a_{22} + T_{y_1} \\ \vdots \\ x'_n = x_n \cdot a_{11} + y_n \cdot a_{12} + T_{x_n} \\ y'_n = x_n \cdot a_{21} + y_n \cdot a_{22} + T_{y_n} \end{cases} \quad (4.35)$$

que através do algoritmo DLT se transformam no sistema matricial

$$\begin{bmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ y_1 & x_1 & 0 & 0 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ y_2 & x_2 & 0 & 0 & 0 & 1 \\ & & \vdots & & & \\ x_n & y_n & 0 & 0 & 1 & 0 \\ y_n & x_n & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \\ T_x \\ T_y \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix} \quad (4.36)$$

Deste sistema matricial resultam os parâmetros que vão definir a matriz de transformação afim H_A, representada assim

$$\boxed{H_A = \begin{bmatrix} a_{11} & a_{12} & T_x \\ a_{21} & a_{22} & T_y \\ 0 & 0 & 1 \end{bmatrix}} \quad (4.37)$$

que pode ser decomposta em

$$H_A = H_R \cdot H_S \cdot H_D \cdot H_T$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \zeta & \alpha & 0 \\ 0 & \frac{1}{\zeta} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \quad (4.38)$$

4.3.6. TRANSFORMAÇÃO PROJETIVA

A transformação projetiva é bastante idêntica à transformação afim, com a única diferença de que, para além de todos os parâmetros de uma transformação afim, corrige ainda os parâmetros de perspetividades horizontal e vertical. Esta é a transformação definida através de uma matriz 3x3 designada de matriz fundamental, $H_P = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$.

Esta transformação acrescenta aos já descritos na última transformação, os parâmetros p_x e p_y responsáveis pela eliminação de perspetiva nas imagens. Na Figura 4.5 mostra-se a ideia de correção que se pretende com este tipo de transformação.

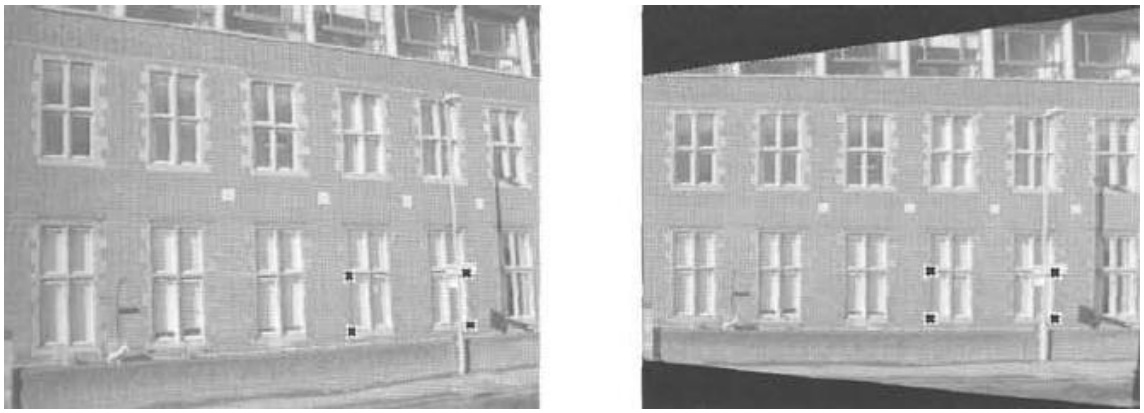


Figura 4.3 – Efeito de uma transformação projetiva estimada por 4 pontos [10].

Na estimação desta transformação são necessários, no mínimo, 4 pontos de cada uma das duas *frames* consecutivas em análise (como mostra a Figura 4.5), pois o número de parâmetros agora a estimar são 8 (com o objetivo de se obter resultados mais estáveis, durante este processo de estimação assumiu-se $h_{33} = 1$, parâmetro arbitrário responsável pelo fator de escala associado às imagens).

Assim, aleatoriamente serão escolhidos 4 pontos de cada conjunto de *keypoints*, sendo calculado, para cada ponto, o sistema de equações que se segue:

$$\begin{cases} x_i' = \dfrac{x_i \cdot h_{11} + y_i \cdot h_{12} + h_{13}}{x_i \cdot h_{31} + y_i \cdot h_{32} + h_{33}} \\ y_i' = \dfrac{x_i \cdot h_{21} + y_i \cdot h_{22} + h_{23}}{x_i \cdot h_{31} + y_i \cdot h_{32} + h_{33}} \end{cases} \qquad (4.39)$$

que em forma matricial se representa por:

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \qquad (4.40)$$

Os passos a seguir para a estimação desta transformação são idênticos aos utilizados para a transformação afim, com a diferença da utilização de mais um ponto para o efeito.

- <u>Primeiro passo:</u>

  O modelo é mais uma vez definido através do cálculo do sistema matricial representado em cima para oito pontos (quatro *matches* de *keypoints*), o equivalente a um sistema de oito equações para oito incógnitas (os oito parâmetros mencionados dizem respeito às incógnitas h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31} e h_{32}, no entanto, os 9 parâmetros serão deduzidos tal como foi feito para a transformação afim).

$$\begin{bmatrix} x_i' \\ y_i' \\ x_j' \\ y_j' \\ x_k' \\ y_k' \\ x_l' \\ y_l' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ x_j \\ y_j \\ x_k \\ y_k \\ x_l \\ y_l \end{bmatrix} \qquad (4.41)$$

É de salientar que a matriz fundamental pode ser, tal como a matriz de transformação afim, decomposta numa transformação afim com acréscimo dos 2 parâmetros de perspetividade, como se apresenta de seguida:

$$H_P = H_A \cdot H_p = H_R \cdot H_S \cdot H_D \cdot H_p$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \zeta & \alpha & 0 \\ 0 & \frac{1}{\zeta} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ p_x & p_y & 1 \end{bmatrix} \qquad (4.42)$$

A ideia passa, mais uma vez, pela estimação da matriz 3x3 que transforma agora os pontos i, j, k e l nos pontos i', j', k' e l' e, por fim, na estimação de cada uma das matrizes que a decompõem. A melhor solução para este sistema de equações está na sua resolução utilizando outra vez o algoritmo DLT. Assim, têm-se oito equações

$$\begin{cases} x_i' = \dfrac{x_i \cdot h_{11} + y_i \cdot h_{12} + h_{13}}{x_i \cdot h_{31} + y_i \cdot h_{32} + h_{33}} \\ y_i' = \dfrac{x_i \cdot h_{21} + y_i \cdot h_{22} + h_{23}}{x_i \cdot h_{31} + y_i \cdot h_{32} + h_{33}} \\ x_j' = \dfrac{x_j \cdot h_{11} + y_j \cdot h_{12} + h_{13}}{x_j \cdot h_{31} + y_j \cdot h_{32} + h_{33}} \\ y_j' = \dfrac{x_j \cdot h_{21} + y_j \cdot h_{22} + h_{23}}{x_j \cdot h_{31} + y_j \cdot h_{32} + h_{33}} \\ x_k' = \dfrac{x_k \cdot h_{11} + y_k \cdot h_{12} + h_{13}}{x_k \cdot h_{31} + y_k \cdot h_{32} + h_{33}} \\ y_k' = \dfrac{x_k \cdot h_{21} + y_k \cdot h_{22} + h_{23}}{x_k \cdot h_{31} + y_k \cdot h_{32} + h_{33}} \\ x_l' = \dfrac{x_l \cdot h_{11} + y_l \cdot h_{12} + h_{13}}{x_l \cdot h_{31} + y_l \cdot h_{32} + h_{33}} \\ y_l' = \dfrac{x_l \cdot h_{21} + y_l \cdot h_{22} + h_{23}}{x_l \cdot h_{31} + y_l \cdot h_{32} + h_{33}} \end{cases} \qquad (4.43)$$

que através do algoritmo DLT se transformam no sistema matricial

$$\begin{bmatrix} -x_i & -y_i & -1 & 0 & 0 & 0 & x_i \cdot x_i' & y_i \cdot x_i' \\ 0 & 0 & 0 & -x_i & -y_i & -1 & x_i \cdot y_i' & y_i \cdot y_i' \\ -x_j & -y_j & -1 & 0 & 0 & 0 & x_j \cdot x_j' & y_j \cdot x_j' \\ 0 & 0 & 0 & -x_j & -y_j & -1 & x_j \cdot y_j' & y_j \cdot y_j' \\ -x_k & -y_k & -1 & 0 & 0 & 0 & x_k \cdot x_k' & y_k \cdot x_k' \\ 0 & 0 & 0 & -x_k & -y_k & -1 & x_k \cdot y_k' & y_k \cdot y_k' \\ -x_l & -y_l & -1 & 0 & 0 & 0 & x_l \cdot x_l' & y_l \cdot x_l' \\ 0 & 0 & 0 & -x_l & -y_l & -1 & x_l \cdot y_l' & y_l \cdot y_l' \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} x_i' \\ y_i' \\ x_j' \\ y_j' \\ x_k' \\ y_k' \\ x_l' \\ y_l' \end{bmatrix} \qquad (4.44)$$

Da resolução deste sistema extraem-se os parâmetros que definem o modelo a seguir. Para se obter cada um dos 9 parâmetros mencionados para este modelo, procedem-se aos cálculos matriciais e outros matemáticos efetuados para a transformação euclidiana com acréscimo dos outros dois parâmetros de perspetiva.

- Rotação:

$$\frac{\frac{h_{11}}{h_{33}}}{\frac{h_{21}}{h_{33}}} = \frac{\cos\theta \cdot \lambda_1 \cdot \zeta}{\sin\theta \cdot \lambda_1 \cdot \zeta} = \frac{\cos\theta}{\sin\theta} = \tan\theta \Rightarrow \arctan\frac{\frac{h_{11}}{h_{33}}}{\frac{h_{21}}{h_{33}}} = \theta \qquad (4.45)$$

$$H_R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- Escalamento horizontal e vertical:

  Para estimar os dois parâmetros de escalamento da imagem procede-se da mesma forma que na transformação anterior utilizando o algoritmo SVD.

  A partir deste algoritmo aplicado à matriz afim A é possível extrair os seus valores singulares que são os parâmetros dos escalamentos, λ_1 e λ_2.

$$SVD(A) = SVD\left(\begin{bmatrix} \frac{h_{11}}{h_{33}} & \frac{h_{21}}{h_{33}} \\ \frac{h_{21}}{h_{33}} & \frac{h_{22}}{h_{33}} \end{bmatrix}\right) \Rightarrow H_S = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \qquad (4.46)$$

- Compressão e *Shear*:

  Manipulando os valores h_{11}, h_{12}, h_{21} e h_{22} é possível extrair os restantes parâmetros.

$$\frac{\sqrt{\frac{h_{11}}{h_{33}} \cdot \frac{h_{21}}{h_{33}}}}{\lambda_1} = \frac{\sqrt{(\cos\theta \cdot \lambda_1 \cdot \zeta)(\sin\theta \cdot \lambda_1 \cdot \zeta)}}{\lambda_1} \Rightarrow \zeta \qquad (4.47)$$

$$\frac{\left(\frac{h_{11}}{h_{33}} \cdot \frac{h_{12}}{h_{33}} + \frac{h_{21}}{h_{33}} \cdot \frac{h_{22}}{h_{33}}\right)}{\zeta \cdot \lambda_1^2}$$

$$= \frac{(\cos\theta \cdot \lambda_1 \cdot \zeta)\left(\cos\theta \cdot \lambda_1 \cdot \alpha - \frac{\lambda_2 \cdot \sin\theta}{\zeta}\right) + \left(\sin\theta \cdot \lambda_1 \cdot \alpha - \frac{\lambda_2 \cdot \cos\theta}{\zeta}\right)(\sin\theta \cdot \lambda_1 \cdot \zeta)}{\zeta \cdot \lambda_1^2} \Rightarrow \alpha$$

$$H_D = \begin{bmatrix} \zeta & \alpha \\ 0 & \frac{1}{\zeta} \end{bmatrix} \qquad (4.48) \text{ e } (4.49)$$

- Segundo passo:

  Mais uma vez efetua-se este passo de igual forma ao sucedido no mesmo passo da transformação anterior. Aplicando-se a cada *keypoint* da *frame* I_t uma matriz transformação definida pelos parâmetros do modelo obtendo-se, desta forma, o novo ponto hipótese. Este ponto é subtraído ao seu real correspondente, calculando-se a distância euclidiana entre ambos.

  Para cada *keypoint* calcula-se

$$\begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \qquad (4.50)$$

  sendo $\tilde{P}(\tilde{x}, \tilde{y})$ o valor do ponto estimado.

- Terceiro passo:

  Aqui comparam-se todos os pontos estimados $\tilde{P}(\tilde{x}, \tilde{y})$, sendo calculada a distância euclidiana ao seu real ponto correspondente $P'(x', y')$.

$$Resultado = \tilde{P}^2 - P'^2$$

$$Resultado = (\tilde{x}^2 + \tilde{y}^2) - (x'^2 + y'^2)$$

  Numa situação ideal de um *inlier* seria de esperar que este valor fosse 0 (zero), porém, tal como nas outras transformações, pode existir ruído ou o ponto ser parte de um objeto em movimento, sendo classificado como *outlier*. Assim tem-se:

$$\text{Se } \begin{cases} Resultado < Threshold \\ Resultado > Threshold' \end{cases} \text{ então } \begin{cases} Ponto \text{ é } inlier \\ Ponto \text{ é } outlier \end{cases}$$

  Os passos 2 e 3 são repetidos o número de vezes necessário até ser encontrado o melhor modelo que define a transformação entre as duas *frames* consecutivas I_t e I_{t+1}, ou seja, aquele que apresenta o maior número de *inliers*. Desta forma, segundo o algoritmo RANSAC, este número pode ser significativamente reduzido e com fiabilidade bastante alta através da equação (4.11):

$$N = \frac{\log(1 - 0{,}99)}{\log(1 - (1 - 0{,}5)^4)}$$

$$N \backsimeq 72 \text{ vezes}$$

- Quarto passo

Finalmente, com os *inliers* todos encontrados é possível estimar a transformação entre as duas *frames*. Através do algoritmo DLT para um sistema de equações sobredimensionado com todos os *inliers* encontrados tem-se o sistema de n equações:

$$\begin{cases} x_1' = \dfrac{x_1 \cdot h_{11} + y_1 \cdot h_{12} + h_{13}}{x_1 \cdot h_{31} + y_1 \cdot h_{32} + h_{33}} \\ y_1' = \dfrac{x_1 \cdot h_{21} + y_1 \cdot h_{22} + h_{23}}{x_1 \cdot h_{31} + y_1 \cdot h_{32} + h_{33}} \\ x_2' = \dfrac{x_2 \cdot h_{11} + y_2 \cdot h_{12} + h_{13}}{x_2 \cdot h_{31} + y_2 \cdot h_{32} + h_{33}} \\ y_2' = \dfrac{x_2 \cdot h_{21} + y_2 \cdot h_{22} + h_{23}}{x_2 \cdot h_{31} + y_2 \cdot h_{32} + h_{33}} \\ \vdots \\ x_n' = \dfrac{x_n \cdot h_{11} + y_n \cdot h_{12} + h_{13}}{x_n \cdot h_{31} + y_n \cdot h_{32} + h_{33}} \\ y_n' = \dfrac{x_n \cdot h_{21} + y_n \cdot h_{22} + h_{23}}{x_n \cdot h_{31} + y_n \cdot h_{32} + h_{33}} \end{cases} \qquad (4.51)$$

que através do algoritmo DLT se transformam no sistema matricial

$$\begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1 \cdot x_1' & y_1 \cdot x_1' \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1 \cdot y_1' & y_1 \cdot y_1' \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2 \cdot x_2' & y_2 \cdot x_2' \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2 \cdot y_2' & y_2 \cdot y_2' \\ & & & & \vdots & & & \\ -x_n & -y_n & -1 & 0 & 0 & 0 & x_n \cdot x_n' & y_n \cdot x_n' \\ 0 & 0 & 0 & -x_n & -y_n & -1 & x_n \cdot y_n' & y_n \cdot y_n' \end{bmatrix} \cdot \begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \\ T_x \\ T_y \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix} \qquad (4.52)$$

Deste sistema matricial resultam os parâmetros que vão definir a matriz de transformação projetiva H_P, representada assim

$$\boxed{H_P = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}}$$

que pode ser decomposta em

$$H_P = H_R \cdot H_S \cdot H_D \cdot H_p \cdot H_T \qquad (4.53)$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \zeta & \alpha & 0 \\ 0 & \dfrac{1}{\zeta} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ p_x & p_y & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

As transformações descritas acima aplicam-se *frame* a *frame* ao longo de um vídeo, estimando-se as homografias a cada instante de tempo. No entanto, é necessário entender que ao longo do tempo vai havendo uma acumulação de instabilidade que deve ser corrigida tendo

em conta a posição inicial da primeira *frame*. Ou seja, para que seja feita uma correção mais perfeita a homografia é estimada *frame* a *frame* e acumulada à homografia inicial. Matematicamente, os cálculos efetuados para a estabilização do vídeo foram os seguintes:

Sendo H_{c-1} a homografia correspondente à última transformação corrente aplicada ao vídeo de forma a estabilizá-lo e H_a a homografia correspondente à transformação atual entre as duas últimas *frames*, é efetuada a inversão de cada uma delas e de seguida a multiplicação de ambas resultando numa nova homografia corrente, H_c, a aplicar à última *frame* do vídeo. Tem-se então:

$$\begin{cases} H_{c-1}^{-1} = Inverse(H_{c-1}) \\ H_a^{-1} = Inverse(H_a) \end{cases}$$

$$\Rightarrow H_c^{-1} = H_{c-1}^{-1} \cdot H_a^{-1} \qquad (4.54)$$

5. H.264: UM FUTURO PROMISSOR

A proliferação das técnicas utilizadas em videovigilância no mundo tecnológico é cada vez maior, tornando-se, por isso, cada vez mais urgente optar-se por formas inteligentes de construir novos sistemas que sejam soluções rápidas e eficazes no campo da análise de vídeo [12]. O capítulo que se segue descreve um pouco aquilo que poderá ser um caminho bastante promissor para a análise de vídeo na visão por computador: o domínio da compressão de vídeo digital H.264/AVC.

Este capítulo começa por mostrar como é feito todo o processo desta compressão, passando depois à descrição da obtenção dos vetores de movimento presentes na mesma, abrangendo, finalmente, a ferramenta de baixo nível utilizada para manipular a informação contida nos descritores de movimento e extrair-se os vetores mencionados.

A compressão é uma técnica de redução de largura de banda e armazenamento de um vídeo ou imagem para que o espaço ocupado por este seja menor. Com o passar dos anos estas técnicas têm evoluído significativamente numa ótica de melhorar a relação qualidade/tamanho que cada vídeo ou imagem comprimida apresentam. A técnica H.264 é a apresenta a melhor relação qualidade/tamanho. No que diz respeito a este projeto, a compressão H.264 surge como a chave para o melhoramento da velocidade de processamento do algoritmo de estabilização. Esta nova abordagem ao problema vem eliminar a necessidade de descompressão do vídeo cada vez que se pretende analisar o deslocamento indesejado de uma *frame* aproveitando, para esse efeito, as características provenientes do trabalho efetuado durante a compressão do vídeo.

5.1. CARACTERÍSTICAS DA COMPRESSÃO

Com a standartização da compressão H.264/AVC, uma nova codificação de vídeo começou a emergir na videovigilância e nas mais simples câmaras de vídeo amadoras, ultrapassando, no que diz respeito à eficiência, muitas outras formas de compressão já existentes.

Antes de mais é necessário entender o que é o H.264, como surgiu e quais são os principais objetivos desta compressão. O H.264/AVC (H.264 denominação atribuída pela ITU-T como evolução do H.263, e AVC – *Advanced Video Coding* – denominação atribuída pela ISO/IEC como uma evolução do MPEG-4 (MPEG-4 *part* 10)) é a mais recente compressão *standard* internacional de vídeo desenvolvida pelos grupos ITU-T *Video Coding Experts Group* e ISO/IEC *Moving Picture Experts Group* num projeto em pareceria conhecido como *Joint Video Team* (JVT) [20, 21]. Esta nova compressão de vídeo foi desenvolvida com o objetivo de abranger inúmeras áreas de aplicação, tais como: *broadcast* por cabo, por satélite, *modem*, terrestre, etc; armazenamento interativo em dispositivos magnéticos como DVDs; serviços de *streaming* multimédia; Discos *Blu-Ray*; entre outras.

H.264/AVC foi uma técnica inventada com o objetivo de fornecer, sem aumentar significativamente os custo inerentes, uma boa qualidade de vídeo comprimido com uma reduzida taxa de bits (*low bit rates*) face às anteriores técnicas *standard*, isto é, metade ou menos da taxa de bits presente nas compressões MPEG-2, H.263 ou MPEG-2 *part* 2[8].

Na área de análise de vídeo e de extração de movimento para a estabilização de vídeo foi necessário entender as novas características presentes no H.264/AVC/MPEG-4 *Part* 10 que permitem que este seja um tipo de compressão com inúmeras vantagens face a outras já existentes. A estimação Multi-imagem (*Inter-picture prediction*) é uma das características que torna esta técnica de compressão bastante eficaz: utiliza como referência as *frames* previamente comprimidas, podendo ser utilizadas mais do que 16 *frames* anteriores à atual (comparativamente às outras técnicas que só utilizavam como referência 1 *frame* anterior)[8]. No entanto, é necessário entender que na compressão H.264 as *frames* existentes não são todas iguais, existindo assim 3 tipos diferentes de *frames*:

- *Frames* do tipo I (*Intra-coded frames*):

  As *frames* do tipo I são aquelas que contêm as informações acerca da imagem completa, permitindo a reconstrução da imagem na sua totalidade;

- *Frames* do tipo P (*Predictive frames*):

  As *frames* do tipo P são aquelas que transportam informações acerca das transformações entre *frames* adjacentes do tipo P ou I;

- *Frames* do tipo B (*Bi-Predictive frames*):

  As *frames* do tipo B são aquelas que complementam as informações acerca das transformações das *frames* ao longo do tempo e destinam-se a facilitar transições entre *frames* do tipo P e P ou P e I [14].

São estas as características que servem de base a este projeto, sendo feito o estudo das transformações existentes nas *frames* do tipo P para que fosse possível extrair os MVs destas *frames* e, posteriormente, aplicar no algoritmo já desenvolvido na primeira parte deste trabalho. Visto que a compressão H.264 é uma técnica bastante complexa, este projeto foi essencialmente focado nas transformações existentes nas *frames* do tipo P, deixando de parte as transformações existentes nas *frames* do tipo B. O facto de apenas se focar neste tipo de *frames* não traz erros significativos para a estabilização de vídeo, pois mais de 85% das *frames* de um vídeo H.264 é do tipo P e, de uma forma geral, as *frames* do tipo I não apresentam movimentos não sendo, por isso, interessantes para o estudo da estabilização.

[8] http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC - Wikipedia, *H.264/MPEG-4 AVC*

5.2. FFMPEG E A EXTRAÇÃO DOS VETORES DE MOVIMENTO

Quando se olha para as técnicas de deteção de objetos em movimento presentes na compressão H.264/AVC, existem algumas formas que residem na extração dos vetores de movimento (MV). No entanto, os MVs são criados para otimizar a compressão de vídeo e não para representar, de forma otimizada e real, o movimento de objetos ou das próprias *frames* (movimentos indesejados). Assim, e devido a este fator, os MVs extraídos da compressão H.264/AVC contêm ruído requerendo complexidades adicionais para manipulação e extração dos verdadeiros valores de movimento presentes em cada MB dos vídeos [12]. Os MBs de uma *frame* são agrupados em fatias (*slices*) de imagens como se mostra na Figura 5.1 que se segue.

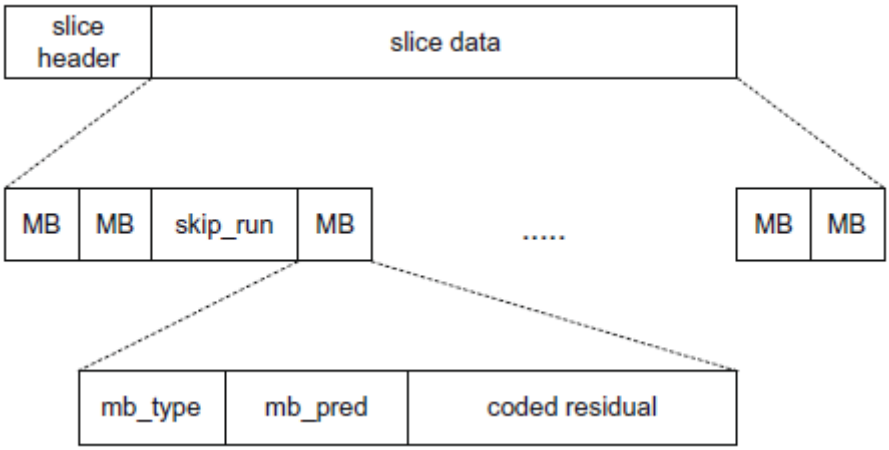


Figura 5.1 – Sintaxe de uma fatia (*slice*) [12].

Como se pode ver nesta figura as fatias são constituídas por um cabeçalho e por um conjunto de informação formado por MBs, cada um contendo informação codificada sobre o tipo, a sua estimação e o resíduo associado. Ora, é neste último termo que reside a informação mais importante para a estabilização de vídeo. Como mencionado anteriormente, os MBs extraídos do H.264 não refletem o verdadeiro movimento das imagens presentes na sequência de *frames* pois contém ruído associado à forma como são estimados. Os vídeos processados em formato H.264 neste trabalho foram, geralmente, vídeos de alta qualidade, contendo por isso um conjunto muito grande de MBs em cada *frame*. Para além disto, cada MB de uma *frame* pode suportar um tamanho de blocos variável, podendo cada um ser dividido em inúmeros blocos, cada um contendo um MV. Somando, no fim pode chegar a ter-se mais de 4000 MV por cada *frame* analisada. Foi, então, necessário manipular esta informação de forma a obter-se o verdadeiro valor do MVs de cada MB e adicionalmente excluir aqueles que continham maior ruído associado.

É neste contexto que entra a ferramenta designada de *FFmpeg*, utilizada para trabalhar diretamente sobre a codificação da compressão H.264. O FFmpeg é uma plataforma computacional desenvolvida e compilada em *Linux* em linguagem *C/C++* para gravação,

conversão e criação de *stream* de áudio e vídeo em diversos formatos. O nome *FFmpeg* provém da extensão do tipo de compressão MPEG em conjunto com a sigla "FF" do inglês *"Fast Forward"* ou do português "avanço rápido". Esta é uma plataforma composta por inúmeras bibliotecas de código *Open Source* capaz de trabalhar em baixo nível sobre qualquer tipo de vídeo. Uma das bibliotecas incluídas nesta plataforma é a biblioteca *libavcodec* responsável pela compressão e descompressão dos *streams* de áudio e vídeo (*av*), tendo sido esta a biblioteca explorada neste projeto para extração das características necessárias para a estabilização de vídeo.

5.2.1. ANÁLISE DO ALGORITMO DESENVOLVIDO

Na biblioteca *libavcodec* foi encontrada toda a informação necessária para se calcular o verdadeiro valor de cada MV presente nos MBs. Nesta biblioteca está presente uma secção de impressão de todos os MV de cada *frame* de um vídeo, porém não existe qualquer tipo de critério para seleção dos melhores MV para descrever o movimento das *frames*. Desta forma, primeiramente excluíram-se MVs iguais a 0, pois correspondem a MB que não contêm qualquer tipo de movimento não sendo necessários para o cálculo das homografias de estabilização.

No H.264 a precisão no cálculo dos MVs é muito mais elevada do que noutras técnicas de compressão. Todos os MVs extraídos de uma *frame* estão presentes num vetor designado de *motion_val[ ]* que contém o valor não real dos movimentos presentes em cada MB analisado, tendo por isso que ser manipulados por forma a obter-se o valor real dos mesmos. Ou seja, os MVs obtidos nesta compressão foram anteriormente manipulados utilizando uma técnica de interpolação designada por *Quarter-pixel* (Qpel). Quando no lado do descodificador são recebidos MVs que apontam para locais que não são posições de valor inteiro na grelha de uma *frame*, novos sub-pixéis devem ser calculados utilizando um método de interpolação. No caso do H.264 a precisão utilizada durante o processo de interpolação é superior a um quarto de pixel sendo, por isso, mais preciso do que as outras técnicas de compressão [15].

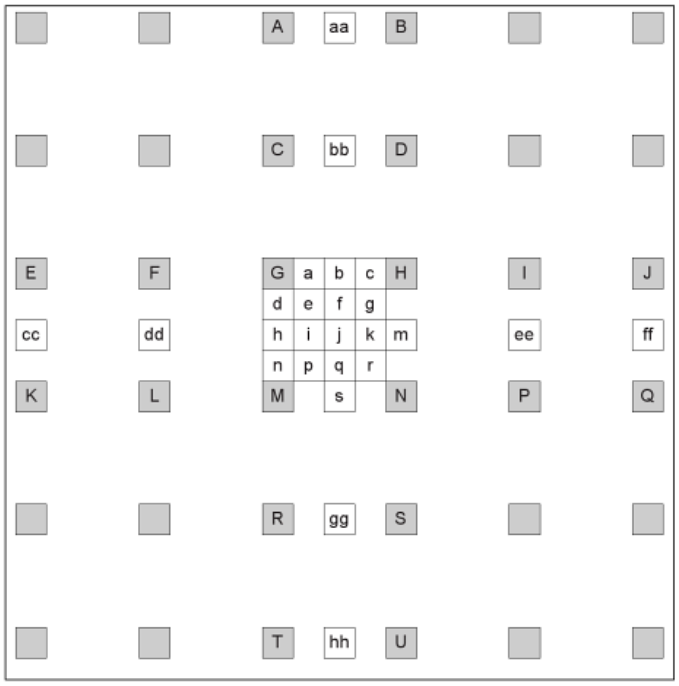


Figura 5.2 – Localização correspondente aos valores inteiros, metades e sub-pixéis na codificação H.264/AVC [15].

Nesta figura ilustram-se os detalhes do processo de interpolação, mostrando-se a cinzento os locais correspondentes a valores inteiros e a branco os locais dos pixéis interpolados com uma precisão de sub-pixel. Os valores presentes no vetor *motion_val[ ]* são, assim, representações de pontos-fixos (*fixed-point representations*) destes valores fracionais retirados da codificação.

Finalmente, estes valores foram convertidos na sua representação fracionária que corresponde ao verdadeiro valor do movimento desse MB de uma *frame*. Geralmente, o fator de escala utilizado para transformar estes valores são potências de 10 ou de 2. No caso em estudo todos os valores foram manipulados com um fator de escala de 2 sendo, por isso, divididos por potências de base 2, 2^n.

A partir deste momento, todos os verdadeiros MVs foram armazenados sendo depois aplicados no algoritmo de estabilização utilizado anteriormente. Para o H.264 foi apenas possível efetuar o estudo da estabilização translacional dos vídeos, ficando a faltar todas as outras transformações possíveis de aplicar para a estabilização. Para estes casos, será necessário estudar com mais pormenores os MVs calculados e manipulá-los por forma a obter valores de rotação e outros em cada *frame*.

6. ANÁLISE DE RESULTADOS

O capítulo que se segue apresenta os resultados obtidos na estabilização de um vídeo capturado com uma câmara HD.

A primeira parte começa por apresentar a análise dos vídeos estabilizados para cada transformação, extraindo-se os dados de cada homografia aplicada *frame* a *frame* ao longo do tempo. A segunda parte efetua um algoritmo de *Benchmarking* entre cada uma das 4 transformações aplicadas ao vídeo, com o objetivo de encontrar aquela que melhor se adapta à situação descrita neste vídeo. Para finalizar é analisado o tempo de processamento do vídeo em estudo. Os tempos calculados dizem respeito à utilização de uma *Central Processing Unit* (CPU), de uma *Graphic Processing Unit* (GPU) e, finalmente, da técnica de compressão H.264 descrita no capítulo 5.

6.1. ANÁLISE DOS PARÂMETROS DE ESTABILIZAÇÃO

Neste subcapítulo apresentam-se os resultados obtidos para cada uma das transformações aplicadas no mesmo vídeo. Começa-se então pela transformação translacional que estabiliza apenas os parâmetros de translações horizontal e vertical.

6.1.1. TRANSFORMAÇÃO TRANSLACIONAL (CPU/GPU)

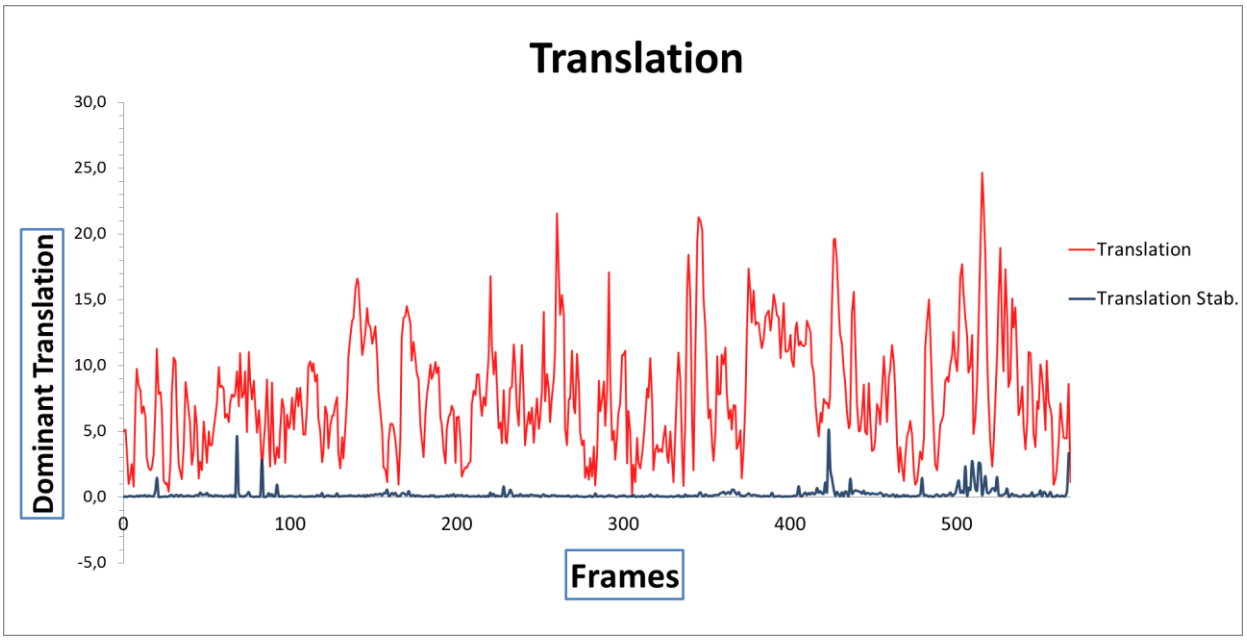


Figura 6.1 – Transformação Translacional: análise de translação (CPU/GPU).

Na Figura 6.1, que mostra um gráfico com os valores de translação em cada instante ao longo do vídeo, encontra-se representado a vermelho a deslocação de cada *frame* em cada instante de tempo para o vídeo não estabilizado; a azul representa-se a deslocação para o respetivo vídeo estabilizado. Nele constata-se que num vídeo que apresenta uma oscilação de cerca de 25 pixéis em determinadas *frames,* a atenuação ao nível destes valores é praticamente total, sendo que o vídeo estabilizado apresenta uma oscilação máxima de 3 pixéis (valores quase impercetíveis). O módulo da deslocação das *frames* é calculado a cada instante de tempo com os valores das componentes x e y (horizontal e vertical, respetivamente) dos vetores de movimento obtidos através dos parâmetros aplicados ao estabilizador de vídeo. Assim, em cada *frame* tem-se:

$$Dominant\ Translation = \sqrt{T_x^2 + T_y^2} \qquad (6.1)$$

Na figura 6.2 mostra-se a *frame* com maior deslocação a ser estabilizada pelo estabilizador projetado. Como se pode ver não há deslocamento na *frame* no vídeo estabilizado (b).

(a) (b)

Figura 6.2 – Resultados da transformação translacional: (a) *frame* estabilizada; (b) *frame* não estabilizada.

6.1.2. TRANSFORMAÇÃO EUCLIDIANA (CPU/GPU)

Para a transformação euclidiana os parâmetros analisados são a translação e a rotação em cada instante. Assim, tal como se verificou na transformação translacional, através da Figura 6.3 é possível verificar que a deslocação de cada *frame* foi atenuada quase totalmente. Neste caso, mais uma vez, as deslocações não excedem os 4 pixéis obtendo-se um gráfico muito mais constante, reflexo de um vídeo muito estabilizado.

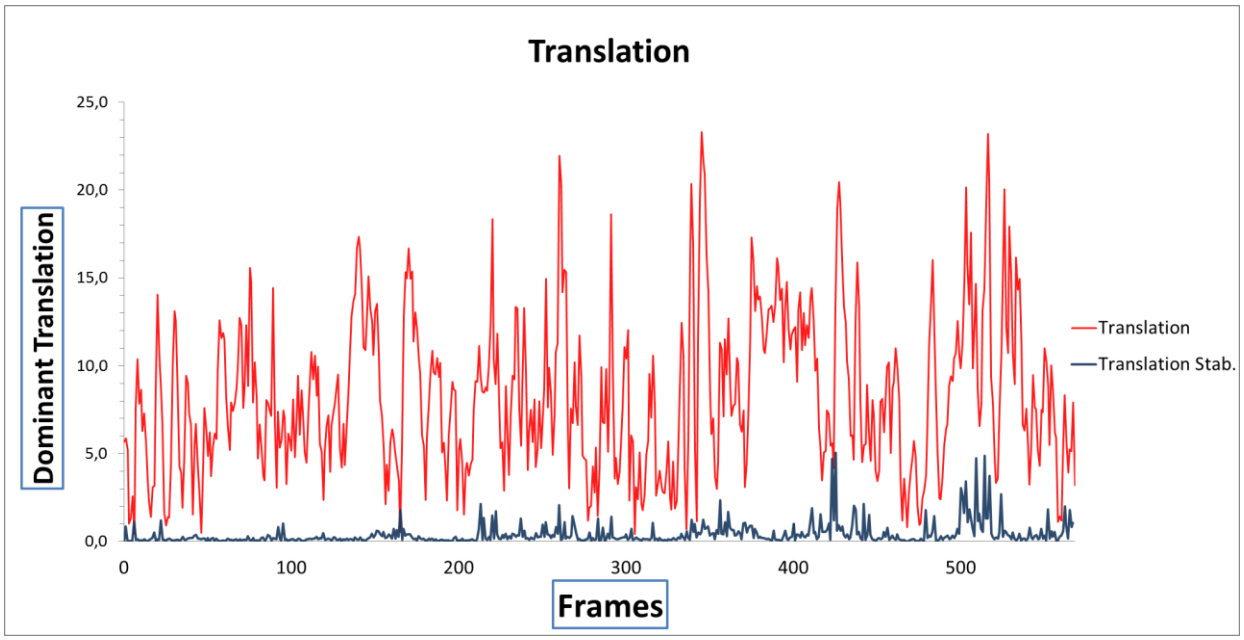


Figura 6.3 – Transformação euclidiana: análise de translação (CPU/GPU).

Da mesma forma que se procedeu na transformação anterior, a translação é calculada através do módulo dos vetores de movimento em cada instante de tempo (equação (6.1)):

$$Dominant\ Translation = \sqrt{T_x^2 + T_y^2}$$

No que diz respeito às oscilações angulares (rotação) observadas no vídeo, os resultados foram igualmente positivos sendo apresentados em seguida na Figura 6.4.

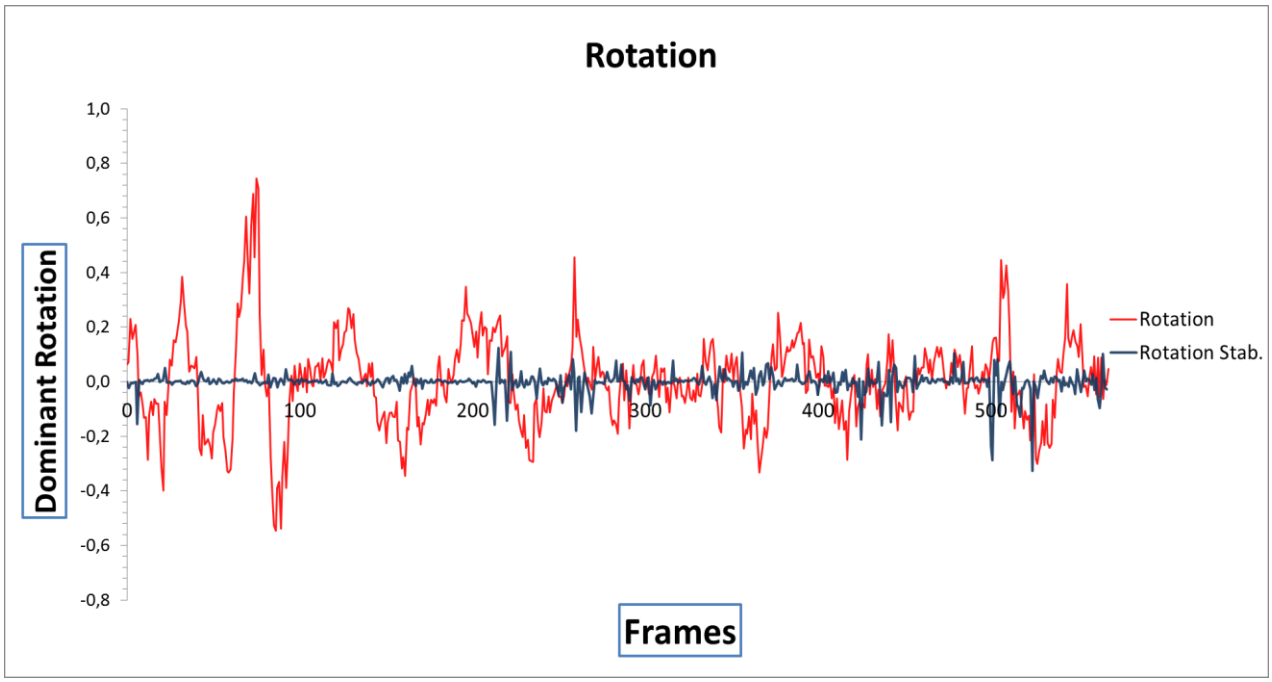


Figura 6.4 – Transformação euclidiana: análise de rotação (CPU/GPU).

Uma vez que se pretende obter valores perto de $0°$, comparando a linha a vermelho com a linha azul constata-se que a atenuação é quase total, deixando de haver perturbações no vídeo estabilizado. Desta forma, passou-se a ter oscilações que não ultrapassam os $0{,}3°$ (valor praticamente impercetível). Esta atenuação pode ser constatada na *frame* 76 apresentada na Figura 6.5, onde se verifica que antes, em (a), se tinha um plano inclinado e agora passou a ter-se um plano direito, em (b). A preto consegue-se ver a correção efetuada nesta *frame*.

(a) (b)

Figura 6.5 – Transformação euclidiana: (a) *frame* não estabilizada; (b) *frame* estabilizada.

A Figura 6.5 (a) mostra o plano horizontal inclinado, enquanto na Figura 6.5 (b) o plano horizontal mantém-se perto dos $0°$.

6.1.3. TRANSFORMAÇÃO AFIM (CPU/GPU)

Como visto no capítulo anterior, a transformação afim estabiliza vários parâmetros para além daqueles já analisados nos dois subcapítulos anteriores. No entanto, estes são parâmetros mais difíceis de observar *frame* a *frame*, sendo só percetíveis numa análise de uma sequência corrida de *frames*. Analisando os dados extraídos pode constatar-se que a estabilização para estes parâmetros mostrou-se, também, bastante eficaz. Começando nos dois parâmetros já analisados nas transformações anteriores, observa-se um comportamento idêntico, tendo-se os mesmos efeitos de estabilidade em ambos os níveis. As Figuras 6.6 e 6.7 mostram os resultados da estabilização aplicada.

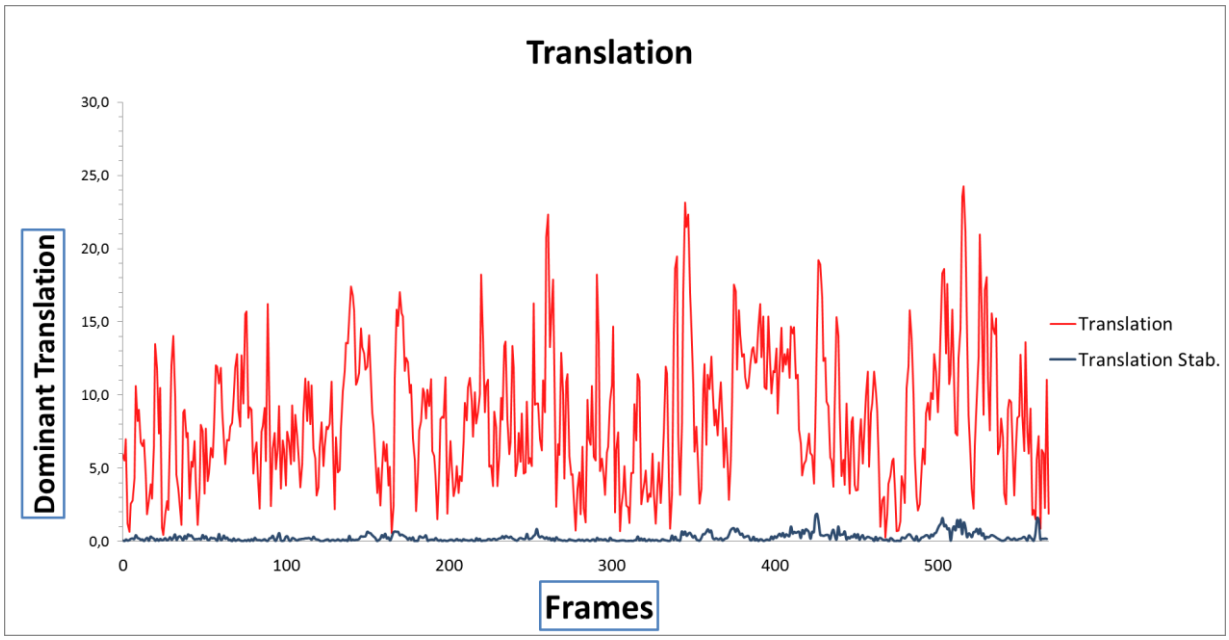


Figura 6.6 – Transformação afim: análise de translação (CPU/GPU).

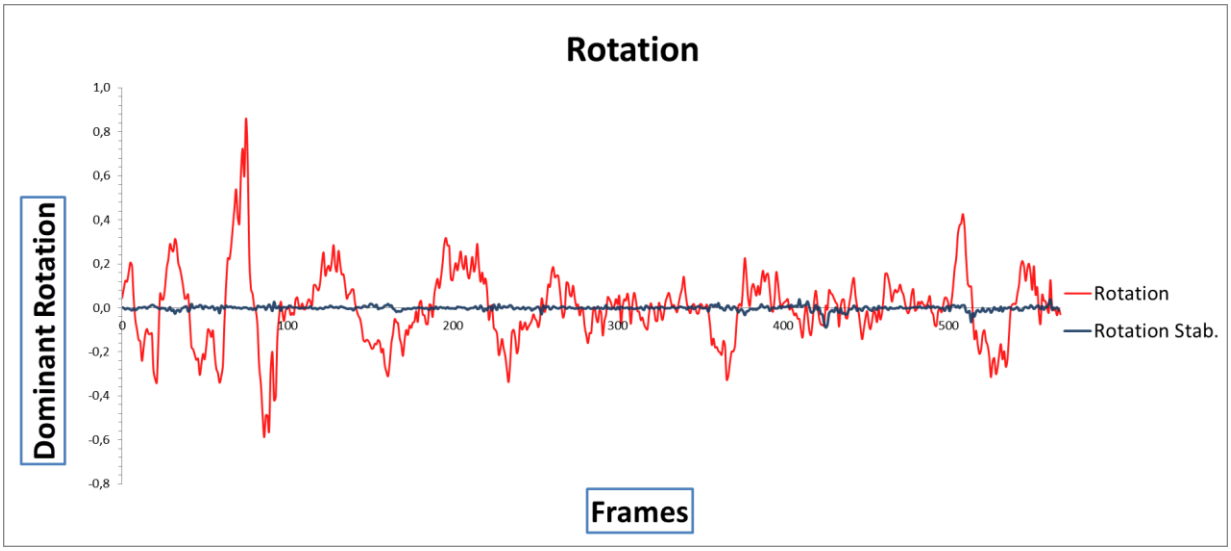


Figura 6.7 – Transformação afim: análise de rotação (CPU/GPU).

Apesar de idêntica às outras, a transformação afim para este vídeo mostra-se ainda mais eficaz pois as oscilações (tanto a níveis translacionais como rotacionais) mantêm-se francamente atenuadas. No primeiro gráfico a linha azul é quase linear em 0 pixéis, refletindo um vídeo extremamente estabilizado em termos de deslocações horizontais e verticais. No segundo gráfico, obteve também uma atenuação muito positiva pois a linha azul mostrou-se totalmente linear em $0°$.

No que diz respeito aos parâmetros *shear*, compressão e escalamento estabilizados por esta transformação, os resultados foram idênticos ilustrando-se nas figuras que se seguem. Primeiramente apresentam-se os dois parâmetros responsáveis pela deformação dos objetos e das próprias *frames*. É de salientar que o *shear* sendo um parâmetro que idealmente deverá ter

o seu valor em 1, na Figura 6.8 apresenta-se com o seu valor logarítmico para melhor se observarem os efeitos da sua estabilização. Assim, nos gráficos este valor deverá idealmente ser 0 (zero). Note-se que tanto o *shear* como a compressão são parâmetros extremamente sensíveis a oscilações, pelo que valores superiores a 0,0015 e 0,005, respetivamente, representam deformações consideráveis de grande instabilidade, devendo ser atenuadas.

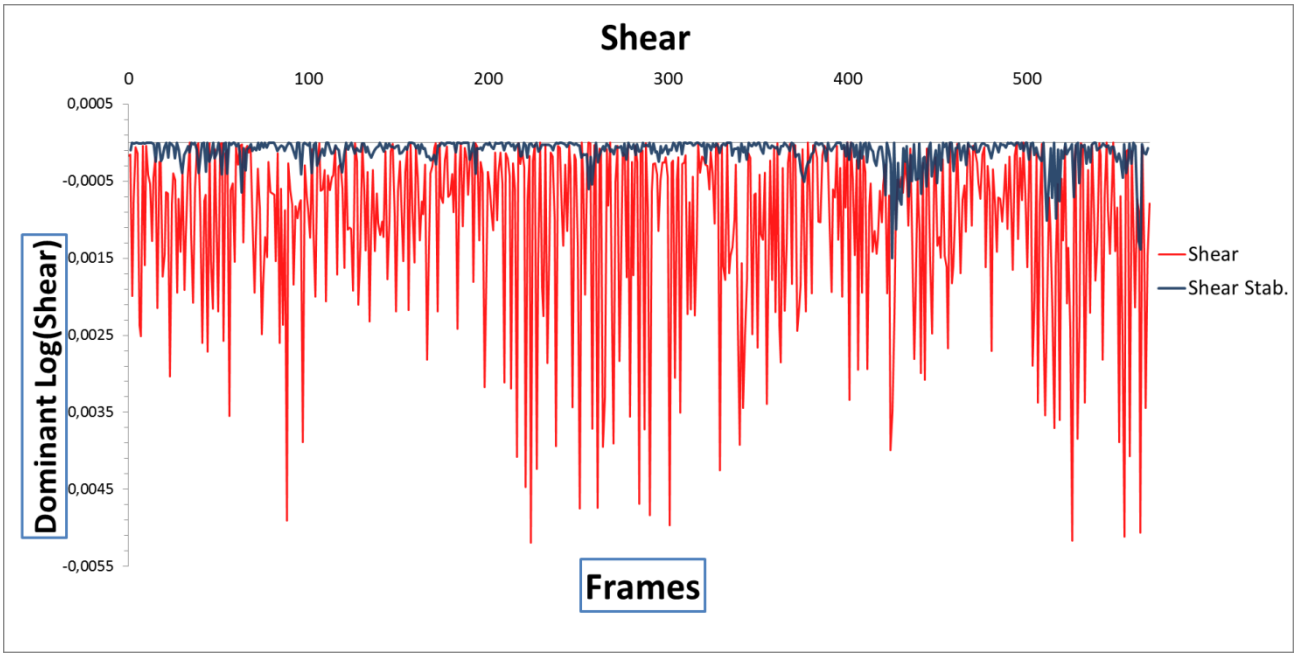


Figura 6.8 – Transformação afim: análise de *Shear* (CPU/GPU).

Como se pode observar a azul, os valores de *Shear*, numa escala logarítmica, não ultrapassam 0.0006 sendo, por isso, considerado muito positivo. O mesmo acontece para a compressão cujo gráfico se ilustra na Figura 6.9.

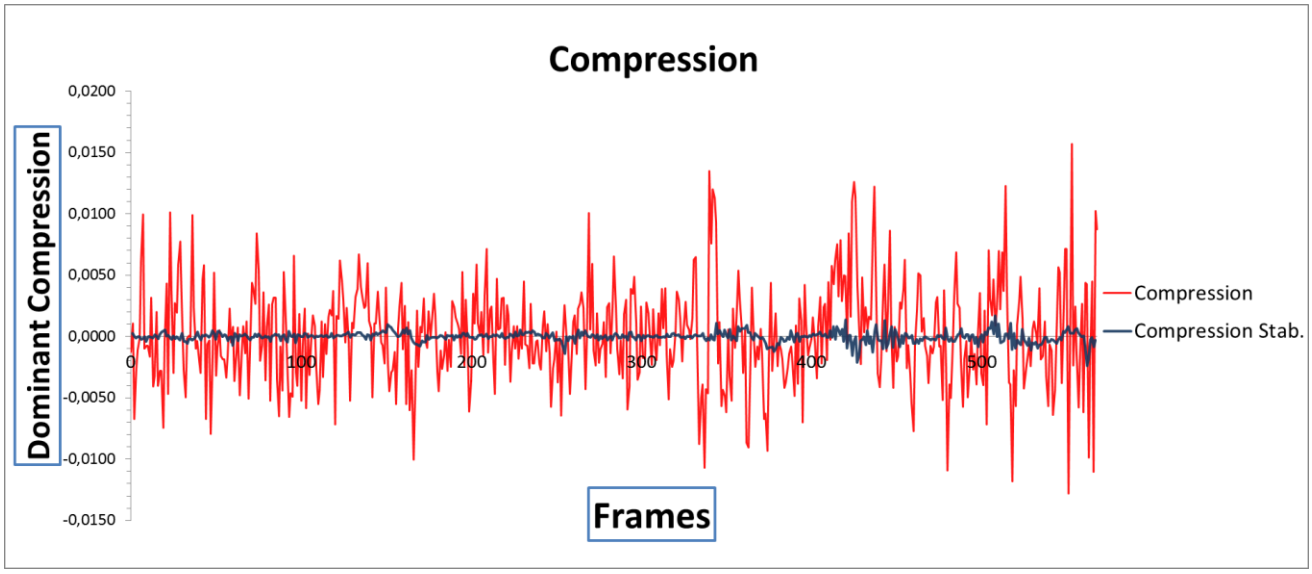


Figura 6.9 – Transformação afim: análise de compressão (CPU/GPU).

Nesta situação observa-se a azul que, tal como seria idealmente espectável para uma situação de um vídeo estabilizado, os valores são praticamente constantes em 0 (zero).

Para finalizar, os resultados que dizem respeito ao escalamento horizontal e vertical apresentam-se nas Figuras 6.10 e 6.11 que se seguem. Estes dois parâmetros são responsáveis pela estabilidade de um vídeo quando existem oscilações em escala, isto é, quando a câmara se desloca indesejavelmente para a frente ou para trás do plano inicial. Assim, estes valores são multiplicativos pelo que deverão, idealmente, situar-se em 1. Por forma a observar-se mais facilmente a atenuação conseguida para estes parâmetros, representam-se nos gráficos os seus valores logarítmicos, pelo que numa situação ideal deverão ser 0 (zero).

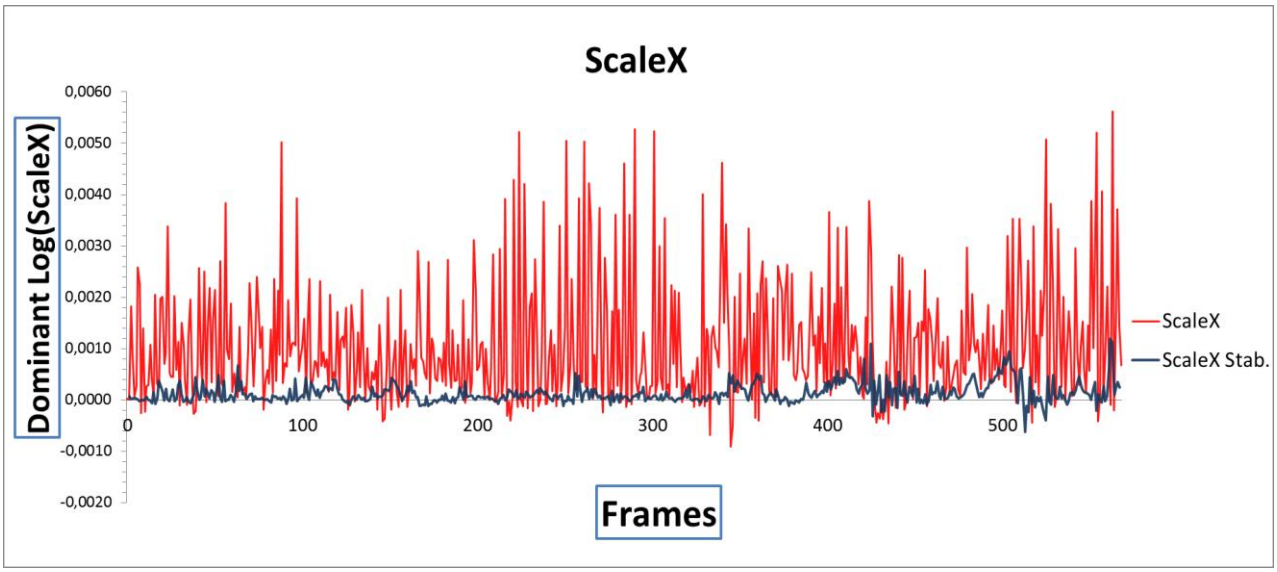


Figura 6.10 – Transformação afim: análise de escalamento horizontal (CPU/GPU).

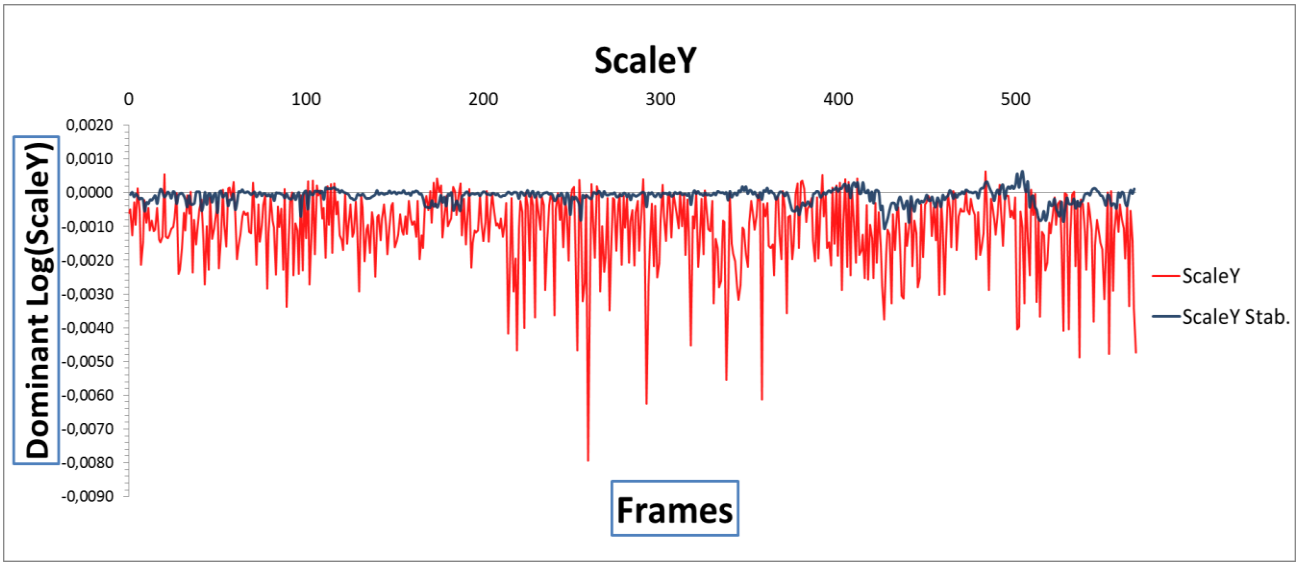


Figura 6.11 – Transformação afim: análise de escalamento vertical (CPU/GPU).

Observando as duas figuras constata-se que se obtiveram resultados tal como os esperados. Em ambas as situações a linha azul é praticamente constante em 0 (zero), refletindo um vídeo bastante estabilizado.

6.1.4. TRANSFORMAÇÃO PROJETIVA (CPU/GPU)

A transformação projetiva é bastante idêntica à transformação afim, com a diferença de que corrige perspetividades numa imagem. De uma maneira geral, obtiveram-se os mesmos resultados que os da transformação anterior, para todos os parâmetros, como é possível observar nas figuras que se seguem.

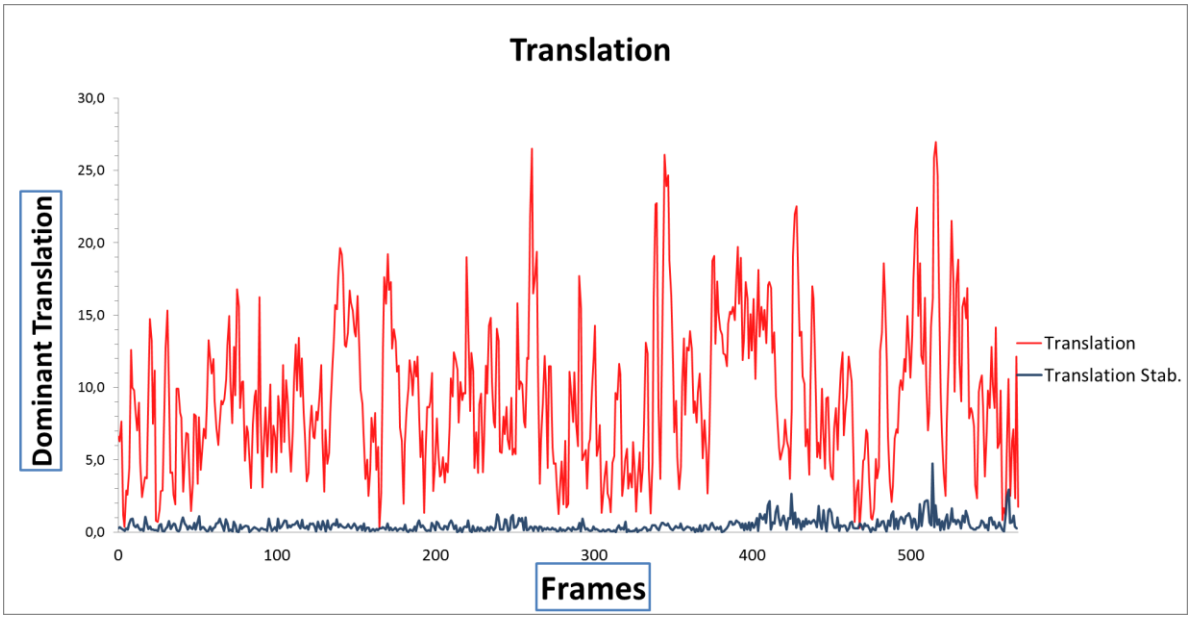


Figura 6.12 – Transformação projetiva: análise de translação (CPU/GPU).

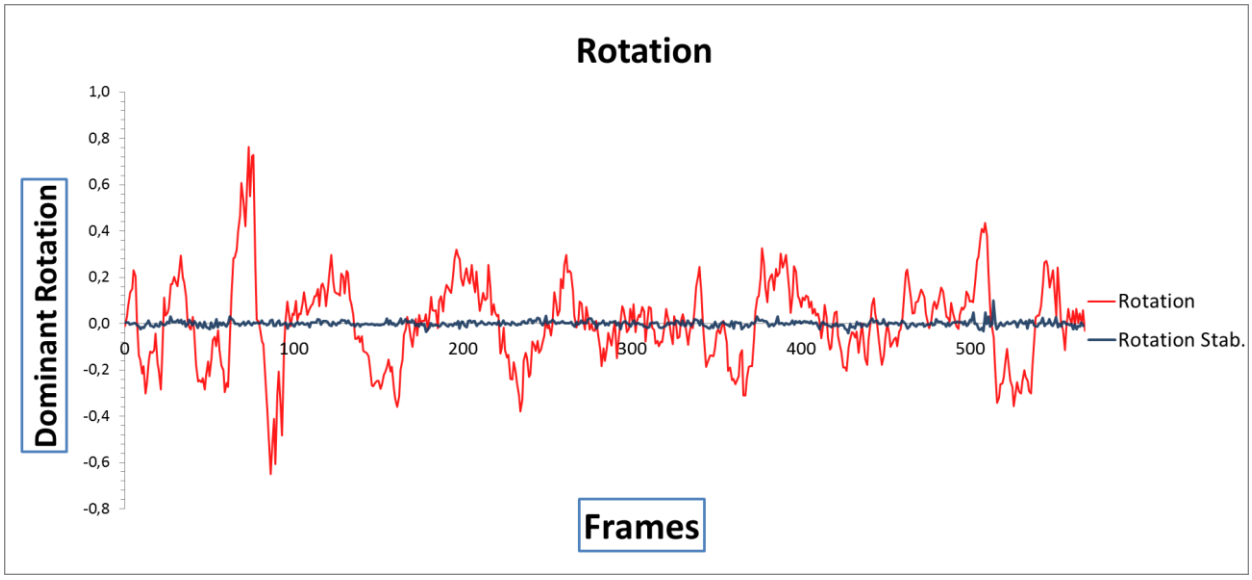


Figura 6.13 – Transformação projetiva: análise de rotação (CPU/GPU).

As Figuras 6.12 e 6.13 mostram como os resultados obtidos nestes dois parâmetros foram extramente positivos pois a linha azul mantém-se constante em torno de 0 (zero), o que reflete uma atenuação das oscilações praticamente total.

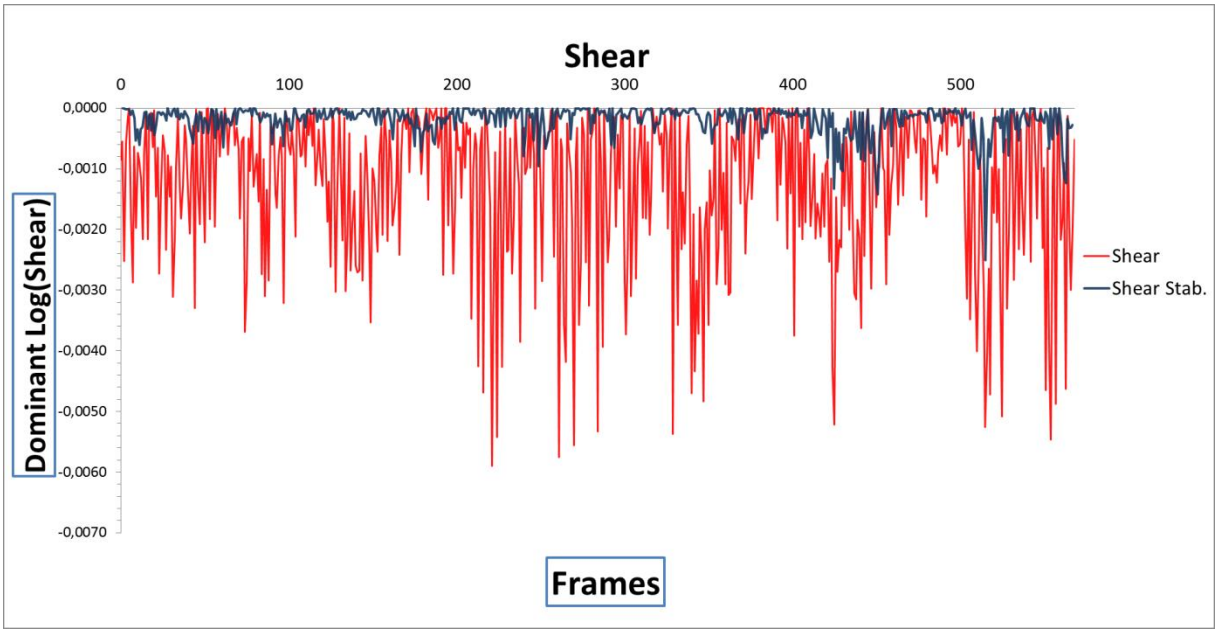


Figura 6.14 – Transformação projetiva: análise de *shear* (CPU/GPU).

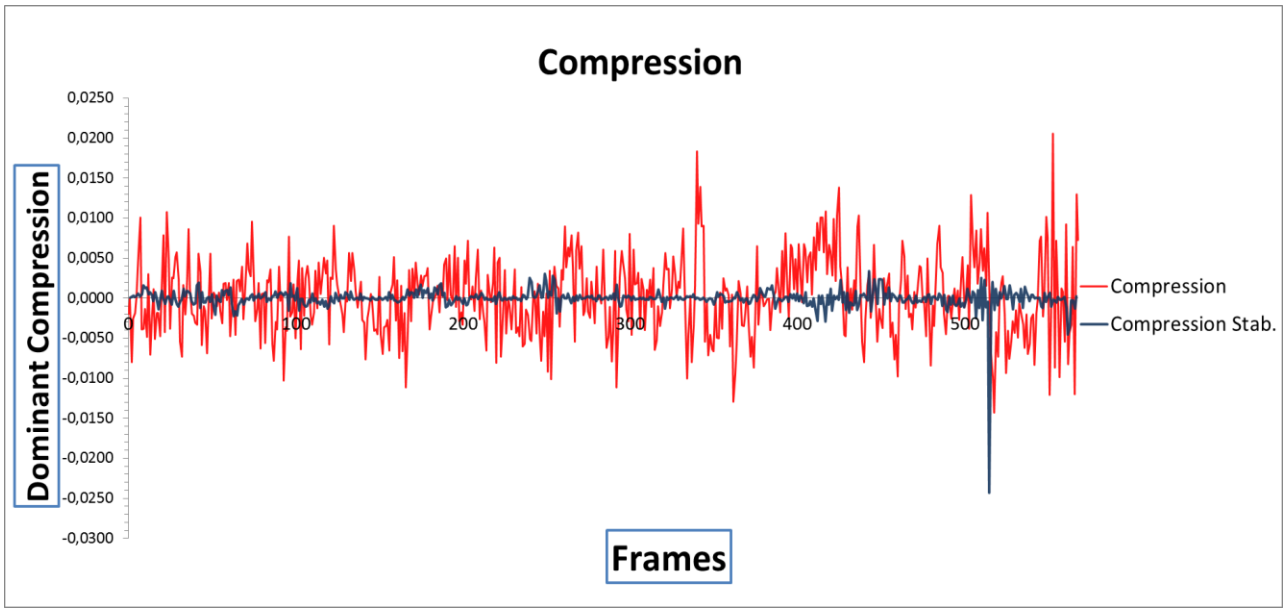


Figura 6.15 – Transformação projetiva: análise de compressão (CPU/GPU).

Os parâmetros de *shear* e compressão foram praticamente atenuados, mantendo os seus valores constantes em 0, tal como seria idealmente esperado. Contudo, na linha azul da Figura 6.15 é observável um "pico" em torno da *frame* 500 que não seria de esperar face aos bons resultados obtidos ao longo do vídeo. Este pico na estabilização deve-se a um erro

associado à separação dos *keypoints* em *inliers* e *outliers*. Na verdade, nesta *frame* o número de *keypoints* foi mais reduzido resultando, assim, num modelo errado para corrigir a instabilidade presente nesta *frame*. A redução do número de *keypoints* nesta *frame* pode dever-se a algum ruído associado ao vídeo naquele instante de tempo provocado por um conceito denominado de *motion blur*. O *motion blur* é um efeito desfocado presente nas imagens causado por movimentos mais rápidos. Este efeito acontece muitas vezes quando existe um movimento muito rápido durante o tempo de exposição da máquina [16].

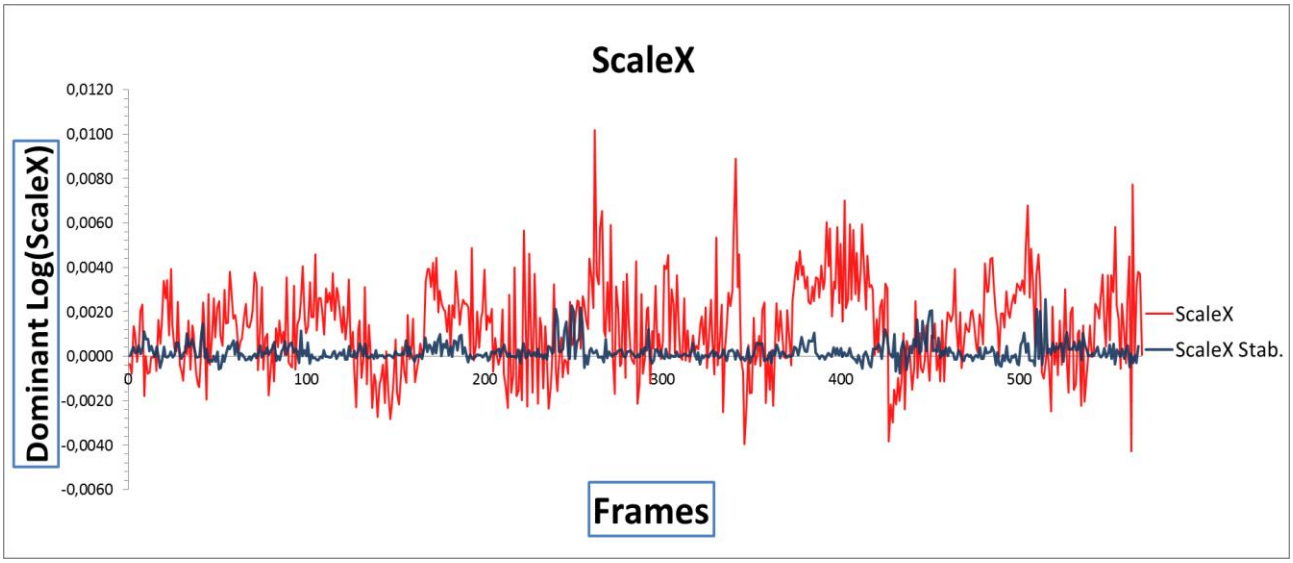


Figura 6.16 – Transformação projetiva: análise de escalamento horizontal (CPU/GPU).

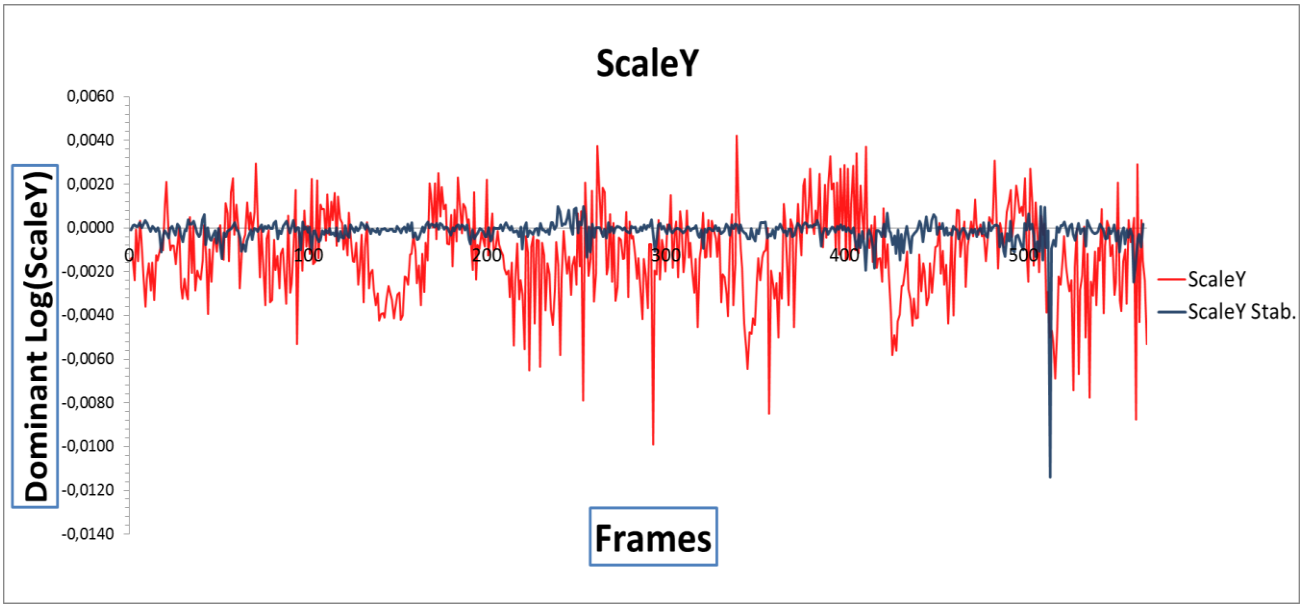


Figura 6.17 – Transformação projetiva: análise de escalamento vertical (CPU/GPU).

Para os valores de escalas horizontal e vertical ilustrados nas Figuras 6.16 e 6.17 os resultados foram igualmente positivos. Tal como numa situação ideal, estes valores

mantiveram-se constantes em 0 (zero) na sua versão logarítmica. Porém, na Figura 6.17 mais uma vez se observa um pico na mesma *frame*, resultado das mesmas razões mencionadas para a Figura 6.15.

Finalmente, nas duas figuras que se seguem, são visíveis os resultados obtidos em relação à correção de perspetividade.

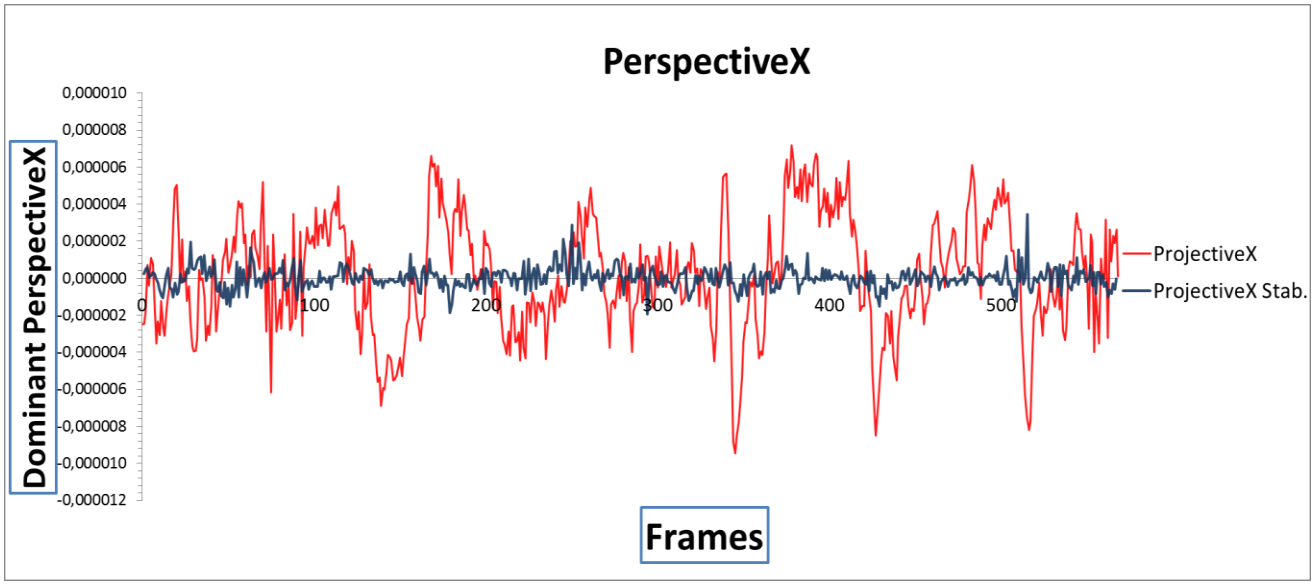


Figura 6.18 – Transformação projetiva: análise de perspetiva horizontal (CPU/GPU).

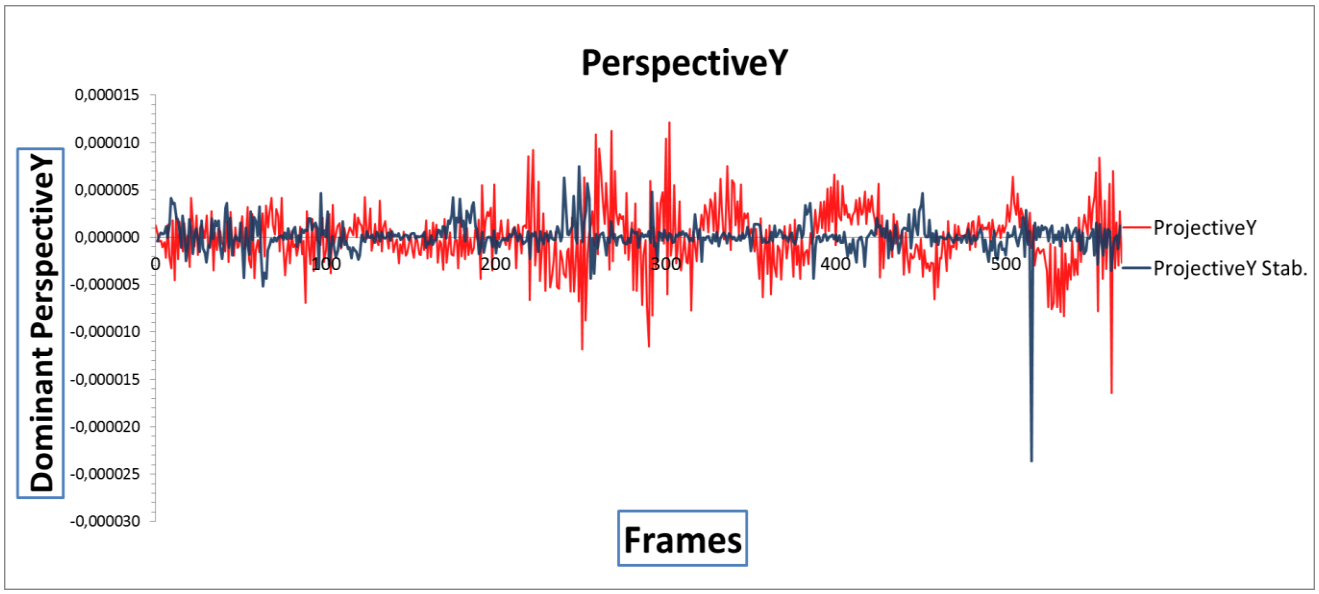


Figura 6.19 – Transformação projetiva: análise de perspetiva vertical (CPU/GPU).

Observando as duas figuras constata-se que, de uma forma geral, os resultados a este nível foram bastante satisfatórios, pois estes valores mantiveram-se constantes em 0 (zero)

como idealmente se projetava. Mais uma vez, em torno da *frame* 500 verifica-se o erro de estabilização já encontrado na correção do escalamento vertical e da compressão.

6.1.5. TRANSFORMAÇÃO TRANSLACIONAL (H.264)

Nesta última análise de parâmetros, apresentam-se os resultados obtidos para a estabilização do mesmo vídeo recorrendo aos seus descritores H.264. Depois de uma análise dos vídeos estabilizados provenientes deste estabilizador, os resultados esperados foram muito positivos. Comparando com a vertente CPU/GPU algumas *frames* não são estabilizadas pois são do tipo B. Estas são *frames* com uma complexidade muito elevada, pelo que não foram analisadas e corrigidas. No H.264 seria de esperar que a atenuação fosse menos precisa, no entanto, os resultados obtidos foram muito bons. Desta forma, conseguiu-se extrair um vídeo bastante estabilizado como se pode observar na análise presente na Figura 6.20.

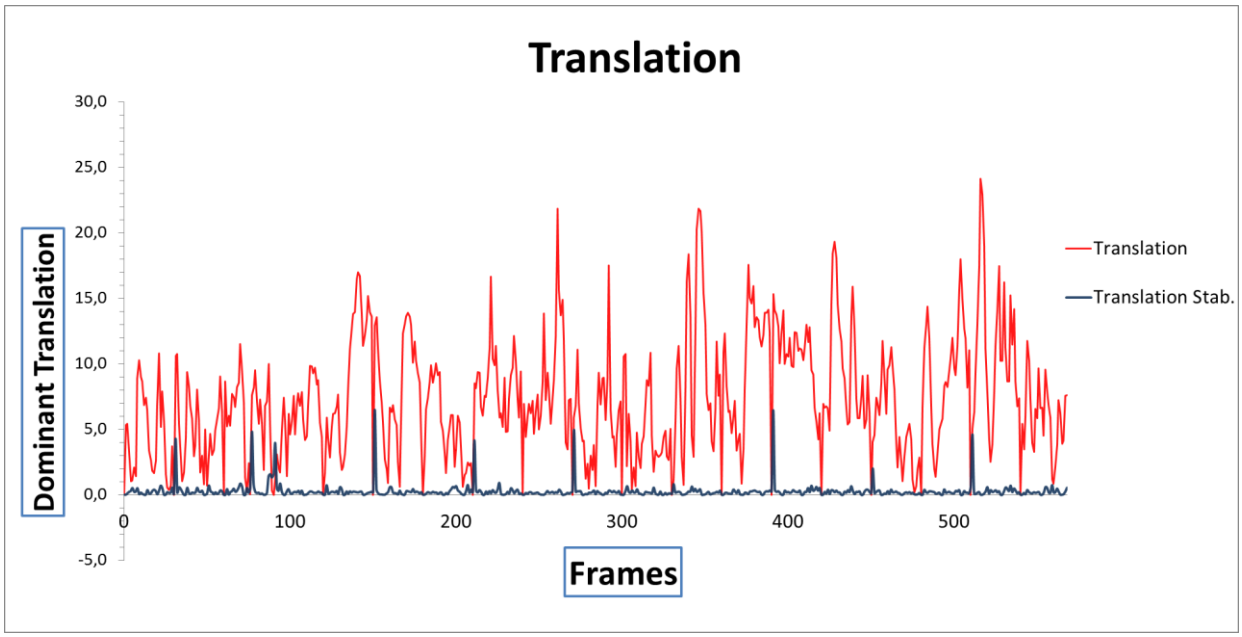


Figura 6.20 – Transformação translacional: análise de translação (H.264)

A linha azul mostra-se constante em 0 (zero) ao longo de todo o vídeo. Porém, notam-se alguns picos ao longo do gráfico pois representam as *frames* do tipo B cuja instabilidade não foi atenuada.

6.2. ANÁLISE TEMPORAL

Observados os resultados obtidos em relação à estabilização obtida através do algoritmo implementado pretende-se, agora, fazer um estudo relativamente ao tempo de execução do estabilizador de vídeo.

Um dos grandes problemas da estabilização digital de vídeo reside no tempo excessivo que este processo leva, não permitindo obter os resultados em tempo real. Desta forma, numa primeira fase analisou-se o comportamento do estabilizador digital aquando da utilização de GPU para processamento paralelo de alguns algoritmos. Numa segunda fase, analisou-se o caso ideal da utilização dos descritores H.264 presentes na compressão de vídeo.

6.2.1. CPU vs GPU

Começou-se então por analisar a performance do estabilizador de vídeo utilizando uma placa gráfica *NVidea* incorporada com uma plataforma denominada de *Compute Unified Device Architecture* (CUDA). A placa utilizada foi a *GeForce GTX 550 Ti* com as seguintes especificações (apenas as mais relevantes)[9] [25]:

- CUDA Cores: 192
- *Graphics Clock* (MHz): 900
- *Processor Clock* (MHz): 1800
- *Memory Clock* (MHz): 4100
- *Standard Memory Config*: 1024 MB GDDR5

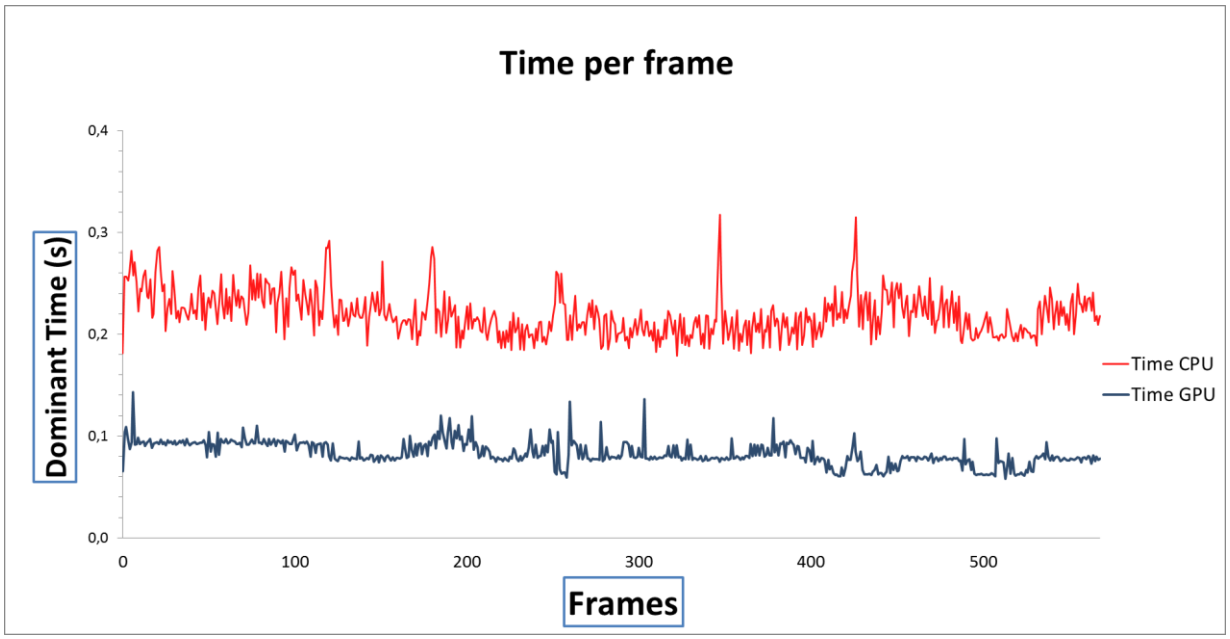
CUDA é uma plataforma de computação paralela e um modelo de programação que permite aumentos significativos de performance computacional ao aproveitar a potência da unidade de processamento gráfico. A utilização deste tipo de *software* tem vindo a crescer e a sua aplicabilidade no mercado é já muito elevada, destacando-se algumas: identificação de placas ocultas em artérias, análise do fluxo de tráfego aéreo, medicina molecular, processamento digital de imagem em diversas áreas, etc[10] [26].

As modificações efetuadas no algoritmo de estabilização para implementação recorrendo ao CUDA dizem respeito apenas às funções pertencentes à biblioteca *OpenCV* utilizada neste projeto. Algoritmos como *Features Detection* (para obtenção dos *keypoints*), *Optical Flow* (para obtenção dos vetores de movimento e, assim, respetivos *keypoints*), conversões de cores, e outros cálculos matemáticos, foram implementados com funções de CUDA pertencentes à biblioteca *OpenCV*.

Olhando para a análise temporal e comparando a performance das duas vertentes, CPU e GPU, constatou-se uma melhoria muito significativa na utilização de uma GPU como auxílio na estabilização de vídeo. Primeiramente estimou-se o tempo associado ao processamento de

[9] http://www.nvidia.com.br/object/product-geforce-gtx-550ti-br.html - NVIDIA, *NVIDIA GeForce GTX-550 Ti*
[10] http://www.nvidia.com.br/object/cuda_home_new_br.html - NVIDIA, *Tecnologia CUDA*

cada *frame* por parte do estabilizador implementado, elaborando-se assim os gráficos presentes nas seguintes figuras. Em cada uma delas comparam-se os tempos para ambas as soluções, CPU e GPU.



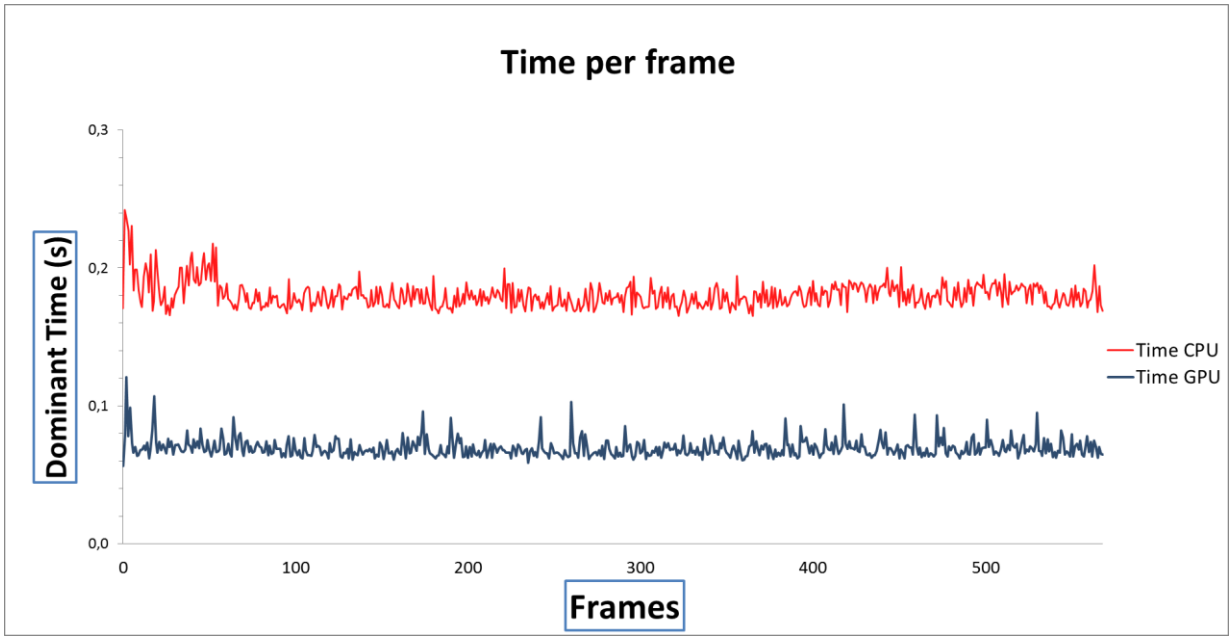


Figura 6.23 – Transformação euclidiana: análise temporal por *frame* (CPU/GPU).

Como se pode observar há uma clara melhoria no tempo de processamento de cada *frame*. Quando antes se tinha um tempo médio de 0,2 s por cada *frame*, passou a ter-se um tempo médio de processamento inferior a 0,1 s.

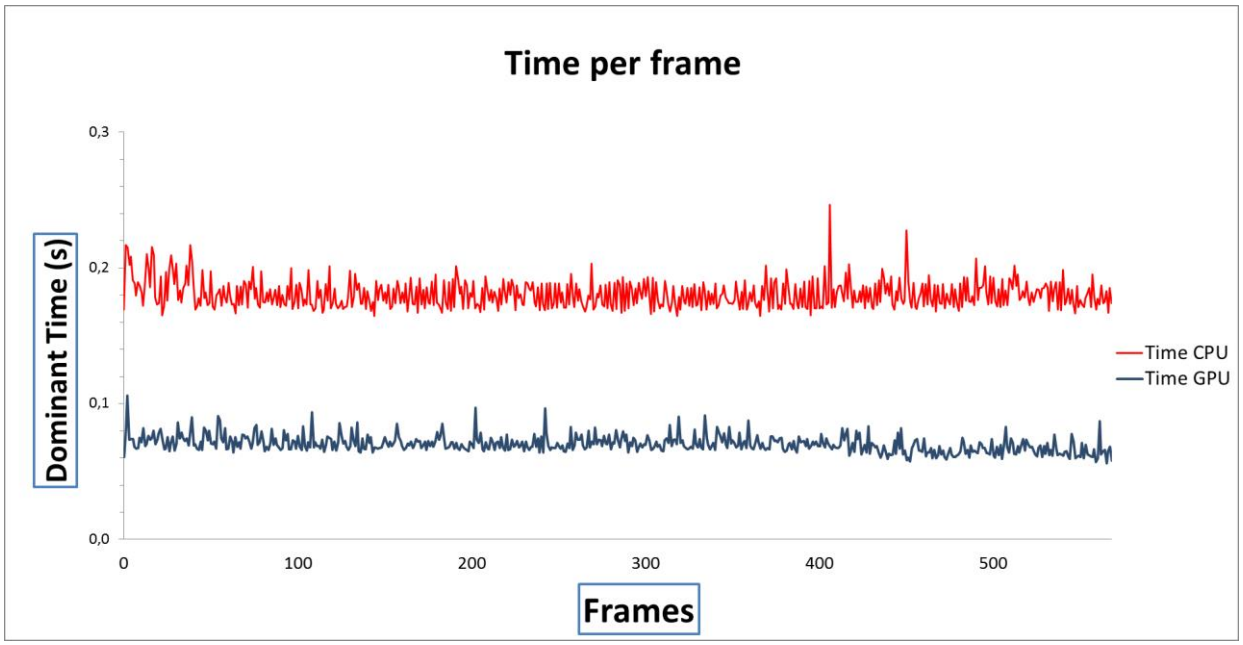


Figura 6.24 – Transformação afim: análise temporal por *frame* (CPU/GPU).

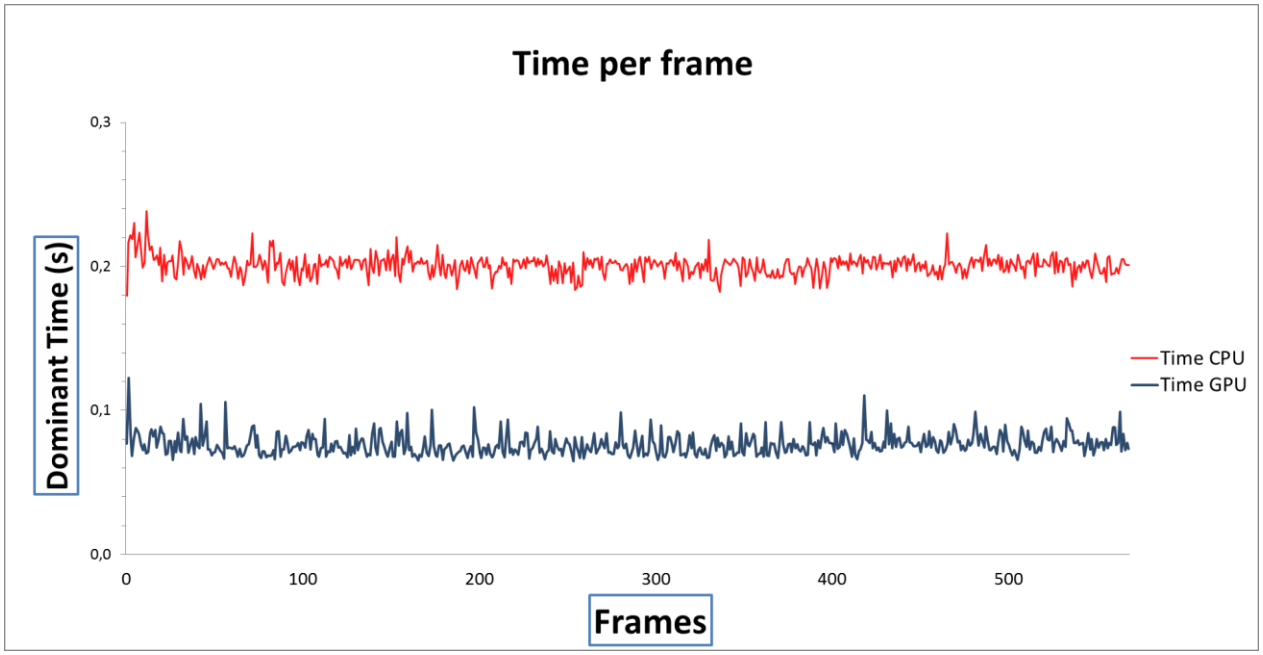


Figura 6.25 – Transformação projetiva: análise temporal por *frame* (CPU/GPU).

Como se pode observar nas figuras, com a utilização da GPU obteve-se uma melhoria bastante significativa em relação à versão que utiliza apenas CPU. Mais uma vez, nestas duas últimas figuras constata-se que houve uma melhoria de mais de metade do tempo de

processamento. Para uma melhor perceção da melhoria calculou-se a média de processamento por *frame* em ambas as situações e estimou-se a percentagem de melhoria de velocidade.

$$Redução = \frac{Média/frame_{GPU} - Média/frame_{CPU}}{Média/frame_{CPU}}$$

Na Tabela 6.1 que se apresenta de seguida podem ser observadas as médias e as melhorias descritas para cada caso.

Transformação	Médias/frame [s]		Tempo total (vídeo 19s) [s]		Melhoria
	CPU	GPU	CPU	GPU	
Translacional	0,219332	0,082865	128,87	54,04	**-62,219%**
Euclidiana	0,180323	0,069293	106,69	42,65	**-61,573%**
Afim	0,180513	0,070153	107,47	43,15	**-61,137%**
Projetiva	0,200626	0,076312	118,72	46,92	**-61,963%**

Tabela 6.1 – CPU *vs* GPU: análise de velocidade de processamento (CPU/GPU)

Tal como seria de esperar, a melhoria de velocidade em cada transformação superou os 60% (redução do tempo de processamento), sendo muito idêntica para cada um destes quatro casos estudados.

6.2.2. CPU vs GPU vs H.264: A SOLUÇÃO IDEAL NA ESTABILIZAÇÃO DE VÍDEO

Para finalizar o estudo temporal efetuou-se a mesma análise temporal feita no subcapítulo anterior, mas agora comparando as versões CPU e GPU com a versão da compressão H.264. Os resultados obtidos foram significativamente positivos conseguindo-se, finalmente, aquilo que não se conseguiu com a utilização de uma GPU de gama média: uma estabilização em tempo real. Na Figura 6.26 apresenta-se o gráfico que mostra a melhoria de velocidade no processo de estabilização. Apenas foi implementada a transformação mais simples sendo, por isso, apenas feita a análise temporal a esse nível.

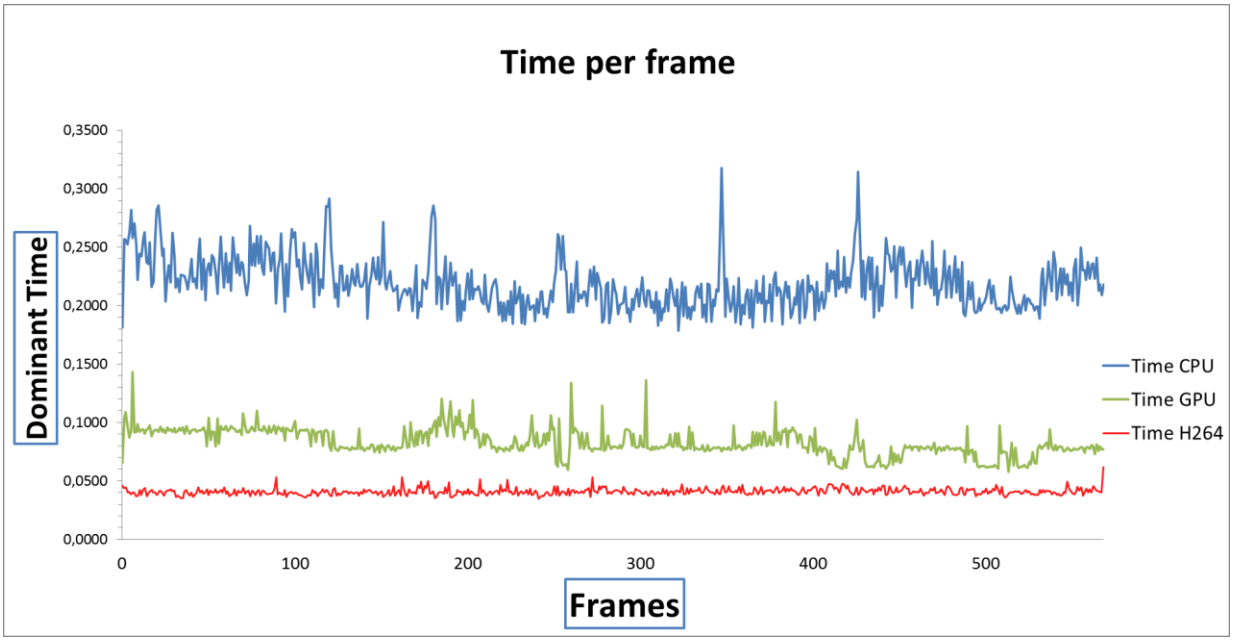


Figura 6.26 – Transformação translacional: análise temporal (CPU/GPU/H.264)

A melhoria de velocidade é notável em mais de metade do tempo face à versão GPU. Numa primeira fase (linha azul – CPU) obteve-se uma velocidade média superior a 0,2 s por cada *frame* estabilizada, sendo que este valor se mostrou bastante oscilante ao longo do vídeo. Na segunda fase (linha verde – GPU) obteve-se uma melhoria significativa, tendo-se valores na ordem dos 0,1 s que, contudo, não resultaram num vídeo estabilizado em tempo real. Finalmente, numa terceira fase (linha vermelha – H.264) os resultados foram extremamente positivos, pois foram alcançados em tempo real e com valores inferiores a 0,05 s, sempre muito constantes em cada *frame* ao longo de todo o vídeo. Na tabela 6.2, mostram-se os cálculos percentuais que comparam a vertente H.264 com as outras duas (CPU e GPU).

Transformação	Médias/frame [s]			Tempo total (vídeo 19s) [s]			Melhoria (CPU/GPU)	Melhoria (CPU/H.264)	Melhoria (GPU/H.264)
	CPU	GPU	H.264	CPU	GPU	H.264			
Translacional	0,219332	0,082865	0,040868	128,8700	54,0400	18,7418	-62,219%	-81,367%	-50,682%

Tabela 6.2 – CPU *vs* GPU *vs* H.264: análise de velocidade de processamento (CPU/GPU/H.264)

Na tabela é constatável a melhoria de velocidade para o H.264 face a qualquer uma das outras opções. Comparativamente à utilização única de CPU a melhoria velocidade com a utilização do H.264 é uma redução de mais de 81% no tempo de processamento. Como se pode ver no tempo total do H.264, o vídeo foi processado em tempo real, tal como era desejável.

6.3. BENCHMARKING (SYMMETRIC TRANSFER ERROR)

Ao longo dos testes efetuados neste projeto surgiu a questão de qual seria, realmente, a melhor transformação a aplicar numa determinada situação de instabilidade. A melhor solução encontrada para avaliar a performance de cada transformação aplicada num mesmo vídeo foi a realização de um *Benchmarking*. O *Benchmarking* é o nome dado a uma forma de executar avaliações comparativas de características de um *software* ou *hardware* recorrendo a testes padrões ou outros ensaios.

No *Benchmarking* utilizado avalia-se o erro final da estabilização, isto é, avalia-se se cada *frame* deslocada voltou à sua posição original tal como seria de esperar. Esta avaliação é feita medindo-se a distância geométrica numa imagem, recorrendo a um método denominado de *Symmetric Transfer Error* (STE). Ou seja, pretende-se calcular a diferença entre o valor medido das coordenadas de uma imagem e o valor estimado das mesmas. Num caso ideal de uma correção perfeita tem-se esta diferença igual a zero.

Começa-se então por medir o erro numa imagem, sendo calculada a distância euclidiana na *frame* I_{t+1} entre o ponto medido x' e o ponto estimado $\tilde{x} = H \cdot x$, onde x é o ponto na *frame* I_t correspondente ao ponto x'. É a esta diferença que se dá o nome de *Transfer Error*. Para finalizar, *Symmetric* é o termo que indica que esta diferença é calculada nos dois sentidos, ou seja, numa situação realista o erro acontece na estimação de ambas as homografias, direta e inversa. Assim, para além do erro geométrico associado à estimação de H, calculou-se também a diferença entre o ponto medido, x, na *frame* I_t e o ponto estimado $\tilde{x} = H^{-1} \cdot x'$ [10]. Desta forma, construiu-se uma função de erro dada pela soma quadrática dos erros geométricos correspondentes a cada uma das homografias por cada *keypoint* das *frames*.

$$\sum_i d(x_i, H^{-1} \cdot x_i')^2 + d(x_i', H \cdot x_i), \qquad (6.2)$$

onde $d(a, b)$ designa a diferença entre a e b.

Assim, foi possível calcular com exatidão aquela que seria a melhor transformação a aplicar em cada *frame* do vídeo por forma a estabilizá-lo o mais corretamente possível.

7. CONCLUSÕES E PERSPETIVAS FUTURAS

Nesta dissertação apresentou-se um sistema de estabilização digital capaz de estabilizar vídeos de alta qualidade com grande precisão e em tempo real, para aplicação no projeto HDA desenvolvido pelo IST em parceria com a empresa Observit. Esta estabilização foi baseada em variações de translação, rotação, escala, compressão, deformação e perspetividade de imagens, com o objetivo de corrigir movimentos indesejados presentes nos mais variados tipos de vídeos, mas sobretudo para aplicação em situações de videovigilância.

Numa primeira fase do projeto de desenvolvimento do algoritmo de estabilização, a estratégia passou por construir um estabilizador capaz de corrigir, de forma bastante robusta, qualquer tipo de movimento indesejado num vídeo, recorrendo-se a métodos e funções presentes na biblioteca *OpenCV* de visão por computador. O que levou ao desenvolvimento de um complexo algoritmo capaz de distinguir os mais variados movimentos indesejados numa câmara. Contudo, dado que o projeto HDA se destina a aplicações reais de videovigilância, foi necessário incluir nesta estratégia a possibilidade da presença constante de objetos em movimento nos vídeos processados. O que levou à adoção de métodos de distinção de movimentos desejados e indesejados, tais como objetos em movimento ou deslocações provocadas pelo movimento da câmara. Desta forma surgiu a necessidade de aplicação do algoritmo RANSAC, por forma a tornar a estimação de homografias de correção de *frames* de vídeos bastante robusta e eficaz.

Testado e verificado o sucesso nos resultados desta estabilização, numa segunda fase deste projeto, surgiu a necessidade de melhorar, significativamente, a performance do estabilizador implementado. Nesta fase desenvolveu-se um novo algoritmo de estabilização idêntico ao anterior recorrendo-se, desta vez, à utilização da tecnologia CUDA incorporada numa GPU *NVidia*. Nesta fase os resultados obtidos em termos de velocidades de processamento foram bastante positivos, tendo sido obtidas reduções em mais de 60% do tempo de processamento inicialmente atingido. Mais concretamente, para um mesmo vídeo com a duração de 20 s, o tempo de estabilização foi reduzido significativamente de 110 para 40 s.

Uma vez concluídas estas duas fases do projeto, os resultados não eram ainda satisfatórios a todos os níveis, pois recorrer a unidades de processamento gráfico de alto nível para melhorar mais a performance do estabilizador poderia não ser a melhor solução numa área cujo investimento é, muitas vezes, elevado. Então, numa terceira e última fase deste projeto, adotou-se uma estratégia de aproveitamento de novas tecnologias presentes na videovigilância e nas mais recentes câmaras de vídeo amador. Já que o maior tempo desperdiçado durante a estabilização estava na descompressão dos vídeos para obtenção dos vetores de movimento presentes em cada *frame*, recorreu-se à nova tecnologia de compressão de vídeo H.264 como uma fonte para obtenção destes vetores. Nesta última fase conseguiu-se

atingir o principal objetivo pretendido para este projeto: a estabilização de vídeos em tempo real.

Em suma, foi um projeto inteiramente bem-sucedido pois todos os objetivos propostos inicialmente foram atingidos na sua totalidade. Para além disto, a aplicação deste projeto no mercado poderá trazer inúmeros benefícios para a videovigilância, no sentido em que reduzirá em grande escala os falsos alarmes recebidos aquando da deteção de movimentos. Falsos alarmes estes que são geralmente gerados a partir de câmaras instaladas em locais sujeitos a perturbações significativas provocadas por ventos fortes.

Numa perspetiva futura, existem ainda muitas melhorias a fazer ao nível da última fase deste projeto respeitante à tecnologia de compressão H.264. Visto que esta é uma tecnologia bastante complexa, há ainda bastante trabalho a realizar no que respeita à obtenção mais precisa dos vetores de movimento extraídos do H.264. Apesar dos resultados em termos de estabilização terem atingido plenamente os seus objetivos, algumas *frames* não foram analisadas, nomeadamente as do tipo B, devido à sua grande complexidade. Por outro lado, existe ainda a necessidade de filtragem do número de MVs que se extraem do H.264. Nesta fase, a exclusão de MVs é fraca, podendo trazer problemas em situações onde existem demasiados objetos em movimento. É, portanto, necessário encontrar formas mais robustas de seleção dos MVs que melhor descrevem o movimento das câmaras. Uma hipótese em vista poderá ser a divisão das *frames* em “super-macro-blocos” e calcular um MV por cada um, reduzindo significativamente o número de MVs utilizados na estabilização dos vídeos.

8. BIBLIOGRAFIA

1. Shakoor, M.H. and M. Moattari. *A new fast method for digital image stabilization*. in *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*. 2010.
2. Ruiming, J., et al. *Digital Image Stabilization Based on Phase Correlation*. in *Artificial Intelligence and Computational Intelligence, 2009. AICI '09. International Conference on*. 2009.
3. Chanda, B. and D.D. Majumder, in *Digital Image Processing and Analysis*, N.-D. Prentice-Hall of India Provate Limited, Editor 2000, Asoke K. Ghosh.
4. Rawat, P. and J. Singhai, *Review of Motion Estimation and Video Stabilization techniques for hand held mobile video.* Signal & Image Processing: An International Journal (SIPIJ), 2011. **2**: p. 10.
5. Brooks, A.C., *Real-Time Digital Image Stabilization.* EE 420 Image Processing Computer Project Final Paper, 2003: p. 10.
6. Jyh-Yeong, C., et al., *Digital image translational and rotational motion stabilization using optical flow technique.* Consumer Electronics, IEEE Transactions on, 2002. **48**(1): p. 108-115.
7. Harris, C. and M. Stephens, *A Combined Corner And Edge Detector.* 1988: p. 6.
8. Burton, A. and J. Radford, eds. *Thinking in Perspective: Critical Issays in the Study of Thought Process*.
9. Lucas, B.D. and T. Kanade, *An Iterative Image Registration Technique with an Application to Stereo Vision.* 1981.
10. Hartley, R. and A. Zisserman, *Multiple View Geometry in Computer Vision*. Second ed. Multiple View Geometry in Computer Vision. Vol. 1. 2003, The Edinburgh Building, Cambridge, United Kingdom: Cambridge University Press.
11. Fischler, M.A. and R.C. Bolles, *Random Sample Consensus: a paradigm for model fitting with applications to image analysis and automated cartography.* Communications of the ACM, 1981.
12. Poppe, C., et al., *Moving object detection in the H.264/AVC compressed domain for video surveillance applications.* Elsevier, 2009.
13. ThomasWiegand, et al., *Overview of the H.264/AVC Video Coding Standard.* 2003. **13**.
14. Zeng, W., et al., *Robust moving object segmentation on H.264/AVC compressed video using the block-based MRF model.* Elsevier, 2005.
15. Campilho, A. and M. Kamel, *Image Analysis and Recognition.* 5th International Conference, ICIAR, 2008.
16. Wikipedia. *Motion Blur*. 2009; Available from: https://en.wikipedia.org/wiki/Motion_blur.